TITLE OF THE INVENTION

VIDEO POKER GAME WITH A BET DOUBLING OPTION

BACKGROUND OF THE INVENTION

Field of the Invention:

[01] The present invention is directed to a method, device, and computer readable storage medium for implementing a video poker game which provides a player with an ability to increase an initial bet, double, or place an additional bet.

Description of the Related Art:

[02] Video poker is a popular gambling game found in casinos.

[03] What is needed is a new variety of the game that can be more profitable for the casino, as well as in a form that some players may prefer over the standard game.

SUMMARY OF THE INVENTION

[04] It is an aspect of the present invention to provide improvements and innovations in video poker games, which increase player enjoyment and casino profitability.

[05] The above aspects can be obtained by a method that includes (a) receiving an initial bet; (b) dealing a first hand of cards to a player; (c) allowing the player to select any number of cards to discard; (d) offering an option for the player to make a second bet; (e) replacing the selected cards to form a final hand; (f) determining a rank of the final hand; (g) paying the initial bet according to the rank; and (h) paying the second bet according to the rank, if the player chose to make the second bet.

[06] The above aspects can also be obtained by a method that includes (a) calculating probabilities for being dealt each rank of a plurality of ranks; and (b) dividing the calculated probabilities by a number of possible paying ranks to obtain payouts for each rank.

The above aspects can also be obtained by a method that includes (a) automatically calculating probabilities for an occurrence of each of a series of events; and (b) automatically dividing the calculated probabilities by a number of events with greater than 0 probability to obtain payouts for each respective event.

[07] The above aspects can also be obtained by a method that includes (a) implementing a video poker game, with the additional feature of allowing a player to place an additional bet after being dealt the initial cards; and (b) paying the additional bet based on a computed paytable based on the player's initial cards.

[08] The above aspects can also be obtained by a computer readable storage that performs (a) implementing a video poker game, with the additional feature of allowing a player to place an additional bet after being dealt the initial cards; and (b) paying the additional bet based on a computed paytable based on the player's initial cards.

[09] The above aspects can also be obtained by a system that includes (a) a processing unit implementing a video poker game, with the additional feature of allowing a player to place an additional bet after being dealt the initial cards; and (b) a paying unit paying the additional bet based on a computed paytable based on the player's initial cards.

[10] These together with other aspects and advantages which will be subsequently apparent, reside in the details of construction and operation as more fully hereinafter described and claimed, reference being had to the accompanying drawings forming a part hereof, wherein like numerals refer to like parts throughout.

BRIEF DESCRIPTION OF THE DRAWINGS

[11] Further features and advantages of the present invention, as well as the structure and operation of various embodiments of the present invention, will become apparent and more readily appreciated from the following description of the preferred embodiments, taken in conjunction with the accompanying drawings of which:

[12] Figure 1 is a flowchart illustrating a basic method of the present invention, according to an embodiment of the present invention;

[13] Figure 2 is a screenshot illustrating a first phase of the invention, according to an embodiment of the present invention;

[14] Figure 3 is a screenshot illustrating the dynamic paytable, according to an embodiment of the present invention;

[15] Figure 4 is a screenshot illustrating a final phase of the invention, according to an embodiment of the present invention;

[16] Figure 5 is a flowchart illustrating a method for computing the dynamic paytable, according to an embodiment of the present invention; and

[17] Figure 6 is a block diagram illustrating one example of hardware that can be used to implement the present invention, according to an embodiment of the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

[18] Reference will now be made in detail to the presently preferred embodiments of the invention, examples of which are illustrated in the accompanying drawings, wherein like reference numerals refer to like elements throughout.

[19] The present invention relates to video poker games and improvements thereof. The present invention provides for a video poker game that allows the player to double (or just increase) his or her bet.

[20] The most common form of video poker found in casinos operates as follows. A player pays to play the game. Five cards are then dealt. The player can then choose to discard any number of the five cards, which are then replaced to form a final hand. A rank of the final hand is determined, and is paid according to a paytable.

[21] To add excitement to the game, the present invention affords the player an option to adjust his bet after the initial cards have been dealt. Of course, some type of alteration

of the game is needed to accommodate this player advantage without the player gaining an advantage over the casino.

[22] In the preferred embodiment of the present invention, the game can be played as the standard game described above. A "double your bet" button is also active after the initial deal and allows the player to optionally double (or just increase) the player's bet after the initial cards have been dealt.

[23] Two paytables can be used. A first paytable is a standard paytable used to pay out the original bet. A second paytable is displayed which automatically adjusts payouts immediately based on cards the player wishes to hold and discard. The payouts are calculated so that the house has an advantage of the house's choosing. If the player chooses the double, then the player's original bet can be paid according to the first paytable and the additional portion of the bet can be paid according to the second paytable.

[24] Figure 1 is a flowchart illustrating a basic method of the present invention, according to an embodiment of the present invention.

[25] The game typically begins with operation 100, where a video poker machine or electronic gaming device (EGD) receives the player's initial bet.

[26] After the bet is received in operation 100, the method proceeds to operation 102 which deals five cards to the player. Of course, other amounts of cards can be used as well, but the standard amount is five.

[27] After the cards are dealt in operation 102, the method proceeds to operation 104 wherein the player selects which cards he or she wishes to discard. This can be done by pointing to the cards on the screen, pressing buttons, etc.

[28] From operation 104, the method proceeds to operation 106 which updates a dynamic paytable based on which cards the player selected to discard (or hold) in operation 104. The dynamic paytable is calculated so typically the house will always have a little advantage regardless of which cards the player chooses to hold. More on this calculation will be discussed below.

[29] From operation 106, the method proceeds to operation 108 which offers a doubling option. The doubling option can be offered by way of a doubling button. The player can choose to press the button and double his or her bet, or just press a standard "draw" button to proceed without doubling.

[30] From operation 108, the method proceeds to operation 110, which then deals replacement cards for the cards which were selected to be discarded to form a final hand.

[31] From operation 110, the method proceeds to operation 112, which accounts for the bets. This is done by determining the rank of the final hand, and paying the initial bet based on a standard paytable. If the player chose to double, then the doubled portion of the bet is paid based on the dynamic paytable.

[32] Figure 2 is a screenshot illustrating a first phase of the invention, according to an embodiment of the present invention.

[33] The first phase as indicated in Figure 2 corresponds to operation 102 from Figure 1. A player has already placed his or her bet of five coins.

[34] A rank list 200 displays winning hand ranks. Paytable 1 202 displays award amounts for the initial bet, for each respective rank from the rank list 200. Paytable 2 204 displays award amounts for an additional bet.

[35] A first coin in display 206 indicates how many coins were bet. A second coin in display 208 also displays how many coins were bet. A balance display 210 displays the player's total balance (how much money he has). A return display 212 displays the computed return for paytable 2 204. It is preferred that the return display 212 not be used in an actual public game, as it may distract players.

[36] A first hand 214 is displayed which is the hand where the player's goal is to make one of the winning hand ranks in the rank list 200. A bet 1 button 216 allows the player to bet 1 coin, and a bet 5 button 218 allows the player to quickly bet 5 coins. The bet 1 button 216 and the bet 5 button 218 are typically not active after the player has already placed his bet. A deal/draw button 220 both allows the player to deal the initial cards after indicating how many coins to bet, and then draw cards after the player has selected the discards. In this phase of the game, the deal/draw button 220 serves the latter operation. A double and draw button 222 allows the player to double the player's bet and then draw cards. At this phase in the game, the deal/draw button 220 and the double and draw button 222 are both active, and the player can choose whether to just draw new cards or double his or her bet and draw new cards.

[37] Paytable 2 204 can also be considered a "dynamic paytable," because it immediately updates the listed payouts based on certain conditions. In this case, when particular cards are selected to be held, the paytable 2 automatically updates the payouts to reflect odds for forming each respective rank. In Figure 2, none of the cards are selected to be held, so paytable 2 204 reflects payouts for discarding and redrawing all 5 cards.

[38] Figure 3 is a screenshot illustrating the dynamic paytable, according to an embodiment of the present invention.

[39] Figure 3 corresponds to operations 104 and 106 from Figure 1, and follows Figure 2. The difference in play from Figure 2 and Figure 3, is the player has selected to hold a queen of clubs 302 and an ace of diamonds 304.

[40] Note that paytable 2 300 has changed from its form from Figure 2 (item 204). This is because these new payouts reflect that the player has decided to hold the queen of clubs 302 and the ace of diamonds 304.

[41] Note that certain payouts are not active in paytable 2 300. For example, royal flush, straight flush, and flush all pay an amount of zero. This is because these hands are impossible to make considering the cards being held. The player can experiment by selecting different cards to hold and viewing the updated paytable 2 300.

[42] Figure 4 is a screenshot illustrating a final phase of the invention, according to an embodiment of the present invention.

[43] Figure 4 corresponds to operation 108 from Figure 1, and is the result of pressing the double and draw button (item 222 from Figure 2) from the state displayed in Figure 3. Since the player doubled his or her initial bet of 5, the new amount of coins bet is 10, which is displayed in the coin bet display 408.

[44] The cards that were not selected to be held are replaced by newly dealt cards to form a second hand 400. Note that the second hand 400 includes a pair of queens, but no other winning combinations, thus this hand is considered to be a "jacks or better" rank.

[45] A rank highlight 402 highlights a winning rank and wining payouts in one or both of paytable 1 404 and paytable 2 406. Paytable 1 404 indicates a winning amount of 5 for the rank of jacks or better. Paytable 2 406 also indicates a winning amount of 5 for the rank of jacks or better. Since the player chose to double his or her bet, the player receives payouts from both paytable 1 404 and paytable 2 406.

[46] Thus, the player wins the sum of the two payouts which is $10, which is displayed in the win display 410. The balance display 412 reflects the win of $10. Since the player wagered $10 and won $10, the player has broke even on this transaction and can now start a new game.

[47] The additional wager is not limited to double of the original bet, but can also comprise any amount of coins (or any fraction) bet the player wishes. Of course, the payouts on the additional bet are based on the number of respective coins bet.

[48] Figure 5 is a flowchart illustrating a method for computing the dynamic paytable, according to an embodiment of the present invention.

[49] The method starts with operation 500, which calculates distributions for each rank. This can be done using a "formulaic" approach. Based on the cards that are selected by the player, and cards remaining in the deck, the number of possible hand of each rank can be determined by counting the number of ways to make each particular rank.

[50] For example, consider a player playing Jacks or Better at the 25 cent coinage level and plays 5 coins. The player is dealt the following cards: 2 of hearts, 4 of spades, 8 of hearts, 9 of clubs, queen of spades. The player decides to hold just the queen of spades. The following is how the number of ways to make each rank can be tabulated formulaically.

[51] Royal flush: The 10, jack, king, and aces of spades are all still in the deck, therefore there is 1 royal flush combination.

[52] Straight flush: The possible spans for a straight flush are 8 to queen and 9 to king. All necessary cards are still in the deck, therefore number of combinations is 2.

[53] Four of a kind: For the ranks 3, 5, 6, 7, 10, jack, king, and ace all four cards are still in the deck, therefore there is one combination each for a total of 8. All three other queens are also still in the deck and the player can still get any of the 44 kickers with the three queens. So the number of four of a kinds is $8 + 44 = 52$.

[54] Full house: The queen can be either part of the three of a kind or pair. If the queen is part of the three of a kind then there are 3 ways to pick 2 queens from the remaining 3. There are 12 ranks left for the pair. 8 of them have all four cards left and 4

have just three left. Of the ranks with all four cards left there are 6 ways to choose 2 cards out of 4. Of the 4 ranks with 3 left there are 3 ways to choose 2 cards out of 3. So the total number of full houses, queens up, is 3*(8*6 + 4*3) = 180. For the number of full houses where the queen is part of the pair there are 3 ways to choose one more queen out of the three left. Of the other 12 ranks there are 4 ways to choose 3 out of 4 cards for the 8 ranks with all four cards remaining. Of the other 4 ranks with 3 cards left there is only 1 way to pick 3 out of 3 cards. So the number of full houses where the queen is the pair is 3*(8*4 + 4*1) = 108. So the total number of full houses is 180+108 = 288.

[55] Flush: Spades are the only possible suit for the flush. The player discarded the 8 of spades so there are 11 spades left in the deck. There are 330 ways to pick 4 spades out of 11 to complete the flush. However 3 of those will result in a straight flush or royal flush. So the number of flush combinations is 330 – 3 = 327.

[56] Straight: There are three possible spans for a straight: 8 to queen, 9 to king, and 10 to ace. The player already discarded an 8 and 9, which will cut down the number of straight combinations. Let n8 = number of 8's left in deck, and so on for each rank. The number of possible straights can be expressed as: n8*n9*n10*nJ + n9*n10*nJ*nK + n10+nJ+nK+nA = 3*3*4*4 + 3*4*4*4 + 4*4*4*4 = 592. However 3 of these combinations result in a straight flush or royal flush. So the final number of straight combinations is 592 – 3 = 589.

[57] Three of a kind: There are two types of three of a kind in this situation: (1) queen is in the three of a kind, (2) queen is a singleton. To determine the number of type (1) three of a kind there are 3 ways to pick 2 out of the three queens left in the deck. There

are also 44 non-queens left in the deck. The number of ways to pick 2 cards out of 44 is 44*43/2 = 946. However we know from the full house section that 8*6 + 4*3 that 60 of these combinations result in a pair. So there are 3*(946-60) = 2658 ways to form a type (1) three of a kind. For the type (2) full houses there are 12 ranks left for the three of a kind, and 11 for the other singleton. The program would circulate through all 132 combinations of three of a kind and singleton ranks. 4*3=12 will result in both ranks having only three cards left, in which case there will be 1*3=3 ways to complete the three of a kind. 8*4=32 ways will result in the 3 of a kind coming from a rank with all 4 cards left and the singleton from a rank with 3. Then there were will be 4*3 = 12 ways to complete the three of a kind. 4*8=32 ways will result in the three of a kind coming from a rank with 3 cards left and the singleton from a rank with 4 cards left. There are 1*4 = 4 ways to complete the three of a kind. 8*7=56 ways will result in both the three of a kind and the singleton coming from ranks with all four cards left. There will be 4*4=16 ways to complete each three of a kind. So the total number of type (2) three of a kinds is (12*3 + 32*12 + 32*4 + 56*16) = 1444. The total number of three of a kinds is 2658 + 1444 = 4102.

[58] Two pair: There are two types of two pairs: (1) queen is part of a pair, (2) queen is the singleton. Of the type (1) two pairs there are 3 possible ranks for the other queen. There are 8*7=56 ways the other pair and singleton can both come from ranks with 4 cards left, for a total of 6*4=24 combinations each. There are 8*4=32 ways the three of a kind can come from a rank of 4 and the singleton from a rank of 3, for a total of 6*3=18 each. There are 4*8=32 ways the three of a kind can come from a rank of 3 and the singleton from a rank of 4, for a total of 3*4=12 combinations each. There are 4*3=12

ways both the other pair and the singleton can come from ranks with 3 left each, for a total of 3*3=9 combinations each. So the total number of type (1) two pairs is 3*(56*24 + 32*18 + 32*12 + 12*9) = 7236. Of the type (2) two pairs there are 8*7/2=28 ways both pairs can come from ranks 4, and there are 6*6 ways to pick the suits from each set. There are 8*4=32 ways to pick one pair from a rank of 4 and one from a rank of 3, and there are 6*3=18 ways to pick the suits from each set. There are 4*3/2=6 ways to pick both pairs from ranks of 3, and there are 3*3=9 ways to pick the suits from each set. So the total number of type (2) two pairs is (28*36 + 32*18 + 6*9) = 1638. The total number of two pairs is therefore 7236 + 1638 = 8874.

[59] Pair: There are two types of pairs: (1) pair of queens, (2) pair of another high card. For the type (1) pairs the program picks one of 3 suits for the other queen and then will cycle through all 12*11*10/6 = 220 ways to pick 3 ranks out of 12 for the singletons. 8*7*6/6=56 of those ways will result in all 3 singletons coming from ranks of 4, for 4^3=64 ways to pick the suits each. (8*7/2)*4=112 of those ways will result in 2 singletons coming from ranks of 4 and one from a rank of 3, for 4^2*3=48 ways to pick the suits each. 8*(4*3/2)=48 of those ways will result in 1 singleton coming from a rank of 4 and two from a rank of 3, for 4*3^2=36 ways to pick the suits. 4*3*2/6=4 ways result from all three singletons coming from ranks of 3, or 3^3=27 ways to pick the suits. So the number of type (1) pairs is 3*(56*64 + 112*48 + 48*36 + 4*27) = 32388 combinations of type (1) pairs. For the type (2) pairs there are 3 ranks to choose from for the other pair. All three ranks have all four cards left so each has 4*3/2=6 ways to arrange the suits. There are 11*10/2=55 ways to pick the ranks of the other two singletons. 7*6/2=21 ways result in both singletons from ranks of 4, for 4^2=16 ways to

pick the suits. 7*4=28 ways result in one singleton from a rank of 4 and one from a rank of 3, for 4*3=12 ways to pick the suits. 4*3/2=6 ways result in both singletons from ranks of 3, for 3^2=9 ways to pick the suits. So the number of type (2) pairs is 3*6*(21*16 + 28*12 + 6*9) = 13068. The total number of pairs is 32388+13068 = 45456.

[60] Non-paying hand: There are 47*46*45*44/24 = 178365 ways to pick 4 replacement cards out of 47 left in the deck. The total number of paying combinations is 59691, adding up the totals for each type of hand. 178365 – 59691 = 118674 ways to have a non-paying hand.

[61] The above method can be implemented when the player holds 0, 1, 2, 3, or 4 cards. Alternatively, a "cycling" method can also be used which deals every possible card combination from the deck and tabulates how many possible ranks can be made. This can be considered a "slow" approach, and is recommended when the player decides to hold 4 cards, thus there are only 47 remaining cards to cycle through and tabulate.

[62] Once the number of possible ways to make each rank is determined based on the cards selected, the method then proceeds to operation 502 which computes a paytable based on the distribution probabilities.

[63] The probability of making each rank can be easily computed by dividing by the number of possible ways to make a rank (computed in operation 500) by the number of possible hands that can be made for the given number of discards. Table I illustrates the number of cards held and how many possible hands can be made.

Table I

# cards held	# hand combinations
0	1,533,939
1	178,365
2	16,215
3	1,081
4	47
5	1

[64] The probability of obtaining a certain rank can be determined by dividing the number of ways to make that rank by the number of possible hands (from Table I). Using the above example where the player is dealt: : 2 of hearts, 4 of spades, 8 of hearts, 9 of clubs, and the queen of spades, and the player decides to hold the queen of spades (which means 4 discards), the following probability table can be computed:

Table II

Rank	# ways to make	probability	1/probability
Royal Flush	1	.0000056	178,571
Straight Flush	2	.0000112	89,286
4 of a Kind	52	.0002915	3,431
Full House	288	.000183	545
Flush	327	.0016	63
Straight	589	.0033	303
3 of a Kind	4,102	.023	43
2 Pair	8,874	.0498	20
Jacks or Better	45,456	.2548	3.9

[65] In Table II, the 1/probability represents a payout for that particular rank, but only if that rank was the only active payout. If a probability for a particular rank is 0, then the payout for that particular rank would be 0 (instead of dividing by 0). Since there are numerous active payouts (9 in Table II), the payouts need to accommodate the others so that overall the paytable does not return more than 100%. To reduce the paytable, the (1/probability) entries can be divided by the number of active paying hands.

[66] For example, in Figure 2 there are 9 active payouts (if a payout is 0 it is not active). So each (1/probability) column can be divided by 9 to result in viable payouts for a game. Table III illustrates the (1/probability) column in Table II divided by 9.

Table III

Rank	1/probability	(1/probability)/9
Royal Flush	178,571	19841
Straight Flush	89,286	9920.667
4 of a Kind	3,431	381.22
Full House	545	60.55
Flush	63	7
Straight	303	33.667
3 of a Kind	43	4.778
2 Pair	20	2.222
Jacks or Better	3.9	.433

[67] The (1/probability)/9 column in Table III represents the payout for each rank for 1 coin bet. This number should be multiplied by the number of coins bet. Further, the payouts in Table III represent no house advantage (due to rounding though there might be a slight house advantage/disadvantage). Typically, a casino would work in a house

advantage so they were guaranteed to make money from the game. Thus, the following formula can be used to obtain a final payout, considering the house advantage and the number of coins bet:

[68] Payout=((coins*game return)/probability of achieving hand)/ # of paying hands

[69] The game return should preferably set to .99 (99%) so that the player would consider the doubling bet a good bet and make it frequently. However, the casino (or game manufacturer) is free to choose whatever game return they wish. Table IV represents the payouts for each rank, and is computed by multiplying Table III by (coins * game return).

Table IV

Rank	((coins*game return)/probability of achieving hand)/ # of paying hands
Royal Flush	98212.95
Straight Flush	49107.30
4 of a Kind	1887.039
Full House	299.72
Flush	34.65
Straight	166.65
3 of a Kind	23.65
2 Pair	11
Jacks or Better	2.14

[70] The paytable in Table IV is a mathematically proper paytable for the above described conditions. However, some adjustments can optionally be made to enhance the player's gambling experience, and to also accommodate casino preferences. Players do not wish to lose their money too quickly. If players lose too quickly, they will be

discouraged and not continue playing or return. Thus, paytables can be shifted to be "bottom heavy." A bottom heavy paytable is one where (probability*payout) for ranks are higher towards the bottom of the paytable than the top. Payouts can be shifted from the higher paying hands (less likely) to the lower paying hands (more likely), while preserving the same overall return for the paytable.

[71] Thus, from operation 502, the method proceeds to operation 504 which shifts payouts.

[72] An algebraic formula can be derived to shift payouts while preserving the same overall return. Table V illustrates an example of a simple paytable.

Table V

Rank	probability	payout
rank1	a	x
rank2	b	y
rank3	c	z

[73] Consider the paytable in Table V to have an even return (1) for simplicity. Now suppose that it is desired to reduce the payout for rank1 by a "shrinking factor" of s, and preserve the same return by increasing rank 2 by a "growth factor" f. Table VI represents what such an adjust paytable would look like.

Table VI

Rank	probability	payout
rank1	a	s*x
rank2	b	f*y
rank3	c	z

[74] If the paytable in Table V has an even return (1), then the following relationship can be stated:

[75] $a*x + b*y + c*z = 1$

[76] The following relationship can be stated from the paytable in Table VI:

[77] $a*s*x + b*f*y + c*z=1$

[78] Using the above two equations and solving for f, we obtain:

[79] $f=(a*x-b*y-a*s*x)/(b*y)$

[80] Thus, if we want to shift 50% of the payout in rank 1 to rank 2, the payout for rank 1 can be multiplied by .50, and to compensate, the payout in rank 2 can be multiplied by f.

[81] In this manner, payouts from the higher paying hands can be transferred to the lower paying hands, to create a more bottom heavy paytable. The source hands, destination hands, and shrinking factor(s) can be set somewhat arbitrarily to suit the designer's preferences.

[82] Optionally, certain dealt hands can be preset to shift payouts in a certain manner. For example, a player is commonly dealt a low pair. When the player is dealt a low pair, the method can automatically shift payouts in a predetermined manner appropriate for the circumstances. For example, 60% of 4 of a kind and 25% of full house can be shifted to two pair and three of a kind, using the shifting methods described above. In this way, the

table is shifted to become more bottom heavy without losing the appeal of attractive payouts on the top.

[83] Further adjustments may still be made to the paytable. In some cases, payouts may be too high. For example, a payout for a royal flush using the above formulas may exceed $100,000, even if a portion is shifted to a lower paying hand as discussed above. Even though casinos will profit from the game in the long run, a casino may be reluctant to offer such large payouts. Therefore, large payouts can be capped and a cap excess can be transferred to lower paying hands.

[84] Thus, from operation 504, the method proceeds to operation 506 which caps selected payouts. High payouts can be optionally reduced to predetermined number(s). The loss in payout due to capping should ideally be shifted to another payout.

[85] For example, if a royal flush pays $55,000 according to the above methods, and the casino or operator wishes to cap this payout at $20,000, then the shrinking factor would be as follows:

[86] s=20,000/ 55,000, or cap amount / current payout

[87] Then, by using the formulas above, the excess amount over the cap can be transferred to another hand, preferably a bottom paying hand. It is recommended that the royal flush be capped and straight flush be capped at a lower amount.

[88] The payouts generated by the above methods will typically contain a fractional part. All of the fractional parts for each payout can simply be removed (such as with an INT() or FLOOR() function), but this will decrease accuracy.

[89] Thus, to improve accuracy, the method proceeds to operation 508, which shifts fractional parts. A preferred method is to shift all of the fractional parts one by one, until no more shifting can be done upon which the fractional part can then be removed. For example, the method can start at the lowest paying payout and shift the fractional part to the next highest payout. From that payout, the fractional part can be shifted again to the next highest payout, and so on, until only the highest payout contains a fractional part. At that point, the fractional part can simply be removed. Since the highest payout is also typically the most unlikely, removing a fractional part of the highest payout would result in the smallest error (deviation from the desired payout return).

[90] In the alternative, instead of shifting the irrational part to the next highest payout, the irrational part can be shifted to the next rank with the next lowest probability, and so on. The shifting of irrational parts can be done using the methods described above.

[91] Once all of the irrational parts are removed, the paytable computing method proceeds to operation 510 which displays the paytable. The above described methods result in generating a dynamic paytable instantaneously.

[92] Further, it is noted the above described methods for automatically generating a paytable are not limited to generating a paytable for a video poker game, but can also be applied to any other game with any type of events. For example, a slot machine can be implemented which instead of using only a fixed paytable, can alter the paytable based on future event probabilities. As yet another example, a dice game can also implement the methods herein, wherein previous rolls of the dice (or other occurrences) can alter probabilities of achieving certain conditions (for example, rolling 4 identical rolls in a

row is more likely after two identical rolls have already been rolled). The invention is further not limited to card games, slot machines, and dice games, but can be applied to any other games with occurring events as well.

[93] Appendix A contains code used to implement the entire game and methods described above. The code is written in the ACTIONSCRIPT language, from MACROMEDIA, which is used to program Flash applications. This language is very similar to C (or C++) and can easily be converted to C or any other programming language. This code is included to provide just one example of how the above method can be implemented, as well as assist one of ordinary skill in the art in implementing the described methods. Of course, many other approaches can be taken as well, in many other different languages.

[94] The routines "fast0", "fast1," "fast2," "fast3," and "slow 4" are the routines that implement the calculating distributions for each rank (operation 500 from Figure 5). Fast0, fast1, fast2, and fast3 implement the formulaic approach, while slow4 implements the cycling approach. The function "adjustable" first calls the appropriate fast or slow routine, and then implements operation 500-508 illustrated in Figure 5 (and the accompanying description) to compute the dynamic paytable. The function "showpays" implements operation 510 illustrated in Figure 5. Other parts of the program are commented and implement the game logic.

[95] It is noted that the dynamic (or second) paytable does not have to include all of the ranks included in the standard paytable used to pay the original wager. The dynamic paytable can also comprise conditions other than ranks indicated in the original paytable,

such as other kinds of hands or other conditions. As only one example, such a condition can comprise a hand that makes up only red (or black) cards. Any such hand or condition can be bet on using the methods described herein. Such conditions can be mixed in any manner with paying hands from the original paytable. Thus, the present invention can provide flexibility in adding a secondary bet to the video poker game (or any other type of game).

[96] It is further noted that any of the games described herein can be played with any kind of deck, either standard or nonstandard. Wildcards can also be used.

[97] In a further embodiment of the present invention, instead of computing paytables on the fly as discussed above, paytables can be precomputed and stored. The paytables can be indexed based on a condition or conditions. When a paytable is desired, based on the condition(s) the proper paytable can be retrieved, displayed, and used.

[98] In a further embodiment of the present invention, the dynamic paytable does not have to be used, but instead a single paytable can be used to pay both the initial and the second bet. This paytable would be modified to accommodate the player advantage of being able to double his bet so that the casino would still have an advantage. Such a paytable can be computed by guessing at such a table, then running through every possible hand with and without doubling, and taking the highest expectation of each. If the expectation is greater than 1, then the paytable payouts can be reduced. However, the method described above using the dynamic paytable is preferred.

[99] In a further embodiment of the present invention, the secondary bet can be mandatory and/or can be paid for by splitting up the initial bet.

[100] In yet a further embodiment of the present invention, the additional bet can be applied to a multi line version of video poker. If additional hands are being dealt, multiple doubled bets can be collected and applied to the multi hands. For example, if three hands are being played at once on a multi line version, the player pays for 3 hands up front, and if he or she wishes to double then the player can pay for three (or any amount) more additional bets.

[101] Figure 6 is a block diagram illustrating one example of hardware that can be used to implement the present invention, according to an embodiment of the present invention. Typically, an electronic gaming device (EGD) is used to implement the present invention.

[102] A processing unit 600 is connected to a ROM 602, RAM 604, and a storage unit 606 such as a hard drive, CD-ROM, etc. The processing unit 600 is also connected to an input device(s) 608 such as a touch sensitive display, buttons, keyboard, mouse, etc. The processing unit 600 is also connected to an output device(s) 610 such as a video display, audio output devices, etc. The processing unit 600 is also connected to a financial apparatus 612, which can accept payments and handle all facets of financial transactions. The processing unit 600 is also connected to a communications link 614 which connects the gaming device to a casino network or other communications network.

[103] It is also noted that any and/or all of the above embodiments, configurations, variations of the present invention described above can mixed and matched and used in any combination with one another. Any claim herein can be combined with any others

(unless the results are nonsensical). Further, any mathematical formula given above also includes its mathematical equivalents, and also variations thereof such as multiplying any of the individual terms of a formula by a constant(s) or other variable.

[104] Moreover, any description of a component or embodiment herein also includes hardware, software, and configurations which already exist in the prior art and may be necessary to the operation of such component(s) or embodiment(s).

[105] The many features and advantages of the invention are apparent from the detailed specification and, thus, it is intended by the appended claims to cover all such features and advantages of the invention that fall within the true spirit and scope of the invention. Further, since numerous modifications and changes will readily occur to those skilled in the art, it is not desired to limit the invention to the exact construction and operation illustrated and described, and accordingly all suitable modifications and equivalents may be resorted to, falling within the scope of the invention.

```
//this program is (c) and not to be used without permission

function setinitialdeck()
{
        indeck = new Array()
for (i=0; i<14; i++) {
        indeck.push (new Array (4));
}

//indeck contains the remaining deck composition used by the fast and slow routines
//the structure is: indeck[rank][suit]
//indeck[13][suit] is all the remaining cards for that suit.
//indeck[rank][4] is all the reminaing cards for that rank
//ranks start at 2 and go through A, i.e. 0=rank of 2, 1=rank of 3, . . . 12=A

indeck[0][0]=1; indeck [1][0]=1; indeck [2][0]=1; indeck [3][0]=1; indeck [4][0]=1;
indeck [5][0]=1;
indeck[6][0]=1; indeck [7][0]=1; indeck [8][0]=1; indeck [9][0]=1; indeck [10][0]=1;
indeck [11][0]=1;
indeck [12][0]=1; indeck [13][0]=13;

indeck[0][1]=1; indeck [1][1]=1; indeck [2][1]=1; indeck [3][1]=1; indeck [4][1]=1;
indeck [5][1]=1;
indeck[6][1]=1; indeck [7][1]=1; indeck [8][1]=1; indeck [9][1]=1; indeck [10][1]=1;
indeck [11][1]=1;
indeck [12][1]=1; indeck [13][1]=13;

indeck[0][2]=1; indeck [1][2]=1; indeck [2][2]=1; indeck [3][2]=1; indeck [4][2]=1;
indeck [5][2]=1;
indeck[6][2]=1; indeck [7][2]=1; indeck [8][2]=1; indeck [9][2]=1; indeck [10][2]=1;
indeck [11][2]=1;
indeck [12][2]=1; indeck [13][2]=13;

indeck[0][3]=1; indeck [1][3]=1; indeck [2][3]=1; indeck [3][3]=1; indeck [4][3]=1;
indeck [5][3]=1;
indeck[6][3]=1; indeck [7][3]=1; indeck [8][3]=1; indeck [9][3]=1; indeck [10][3]=1;
indeck [11][3]=1;
indeck [12][3]=1; indeck [13][3]=13;

indeck[0][4]=4; indeck [1][4]=4; indeck [2][4]=4; indeck [3][4]=4; indeck [4][4]=4;
indeck [5][4]=4;
indeck[6][4]=4; indeck [7][4]=4; indeck [8][4]=4; indeck [9][4]=4; indeck [10][4]=4;
indeck [11][4]=4;
indeck [12][4]=4;
```

```
}



function fast0()
{
var i,j,k,l,numcom;

total[0]=0;total[1]=0;total[2]=0;total[3]=0;total[4]=0;total[5]=0;total[6]=0;
total[7]=0;total[8]=0;total[9]=0;total[10]=0; total[11]=0;

        for (i=0; i<=7; i++)
                for (j=0; j<=3; j++)

        total[10]+=indeck[i][j]*indeck[i+1][j]*indeck[i+2][j]*indeck[i+3][j]*indeck[i+4][
j];

        for (j=0; j<=3; j++)
        {

        total[10]+=indeck[12][j]*indeck[0][j]*indeck[1][j]*indeck[2][j]*indeck[3][j];

        total[11]+=indeck[8][j]*indeck[9][j]*indeck[10][j]*indeck[11][j]*indeck[12][j];
        }

        for (i=0; i<=8; i++)

        total[4]+=indeck[i][4]*indeck[i+1][4]*indeck[i+2][4]*indeck[i+3][4]*indeck[i+4
][4];
        total[4]+=indeck[12][4]*indeck[0][4]*indeck[1][4]*indeck[2][4]*indeck[3][4];

        for (i=0; i<=3; i++)
        {
                if (indeck[13][i]==13)
                        total[5]+=1287;
                else if (indeck[13][i]==12)
                        total[5]+=792;
                else if (indeck[13][i]==11)
                        total[5]+=462;
                else if (indeck[13][i]==10)
                        total[5]+=252;
                else if (indeck[13][i]==9)
                        total[5]+=126;
```

```
                else if (indeck[13][i]==8)
                        total[5]+=56;
                else if (indeck[13][i]==7)
                        total[5]+=21;
                else if (indeck[13][i]==6)
                        total[5]+=6;
                else if (indeck[13][i]==5)
                        total[5]+=1;
        }
        total[4]-=(total[10]+total[11]);
        total[5]-=(total[10]+total[11]);


        if (indeck[12][4]==4) // four aces
        {
                total[9]+=indeck[0][4]+indeck[1][4]+indeck[2][4];

        total[9]+=indeck[3][4]+indeck[4][4]+indeck[5][4]+indeck[6][4]+indeck[7][4]+in
deck[8][4]+indeck[9][4]+indeck[10][4]+indeck[11][4];
        }
        if (indeck[0][4]==4) // four 2's
        {
                total[8]+=indeck[12][4]+indeck[1][4]+indeck[2][4];

        total[8]+=indeck[3][4]+indeck[4][4]+indeck[5][4]+indeck[6][4]+indeck[7][4]+in
deck[8][4]+indeck[9][4]+indeck[10][4]+indeck[11][4];
        }
        if (indeck[1][4]==4) // four 3's
        {
                total[8]+=indeck[12][4]+indeck[0][4]+indeck[2][4];

        total[8]+=indeck[3][4]+indeck[4][4]+indeck[5][4]+indeck[6][4]+indeck[7][4]+in
deck[8][4]+indeck[9][4]+indeck[10][4]+indeck[11][4];
        }
        if (indeck[2][4]==4) // four 4's
        {
                total[8]+=indeck[12][4]+indeck[0][4]+indeck[1][4];

        total[8]+=indeck[3][4]+indeck[4][4]+indeck[5][4]+indeck[6][4]+indeck[7][4]+in
deck[8][4]+indeck[9][4]+indeck[10][4]+indeck[11][4];
        }
        for (i=3; i<=11; i++)
                if (indeck[i][4]==4)
                        total[7]+=43;

        /* full house */
```

```
for (i=0; i<=11; i++)
{
        for (j=i+1; j<=12; j++)
        {
                if ((indeck[i][4]==2)&&(indeck[j][4]==3))
                        total[6]++;
                else if ((indeck[i][4]==3)&&(indeck[j][4]==2))
                        total[6]+=1;
                else if ((indeck[i][4]==2)&&(indeck[j][4]==4))
                        total[6]+=4;
                else if ((indeck[i][4]==4)&&(indeck[j][4]==2))
                        total[6]+=4;
                else if ((indeck[i][4]==3)&&(indeck[j][4]==3))
                        total[6]+=6;
                else if ((indeck[i][4]==3)&&(indeck[j][4]==4))
                        total[6]+=18; /* 4c2+3*4 */
                else if ((indeck[i][4]==4)&&(indeck[j][4]==3))
                        total[6]+=18;
                else if ((indeck[i][4]==4)&&(indeck[j][4]==4))
                        total[6]+=48; /* 2*(4*4c2) */
        }
}


/* three of a kind */
for (i=0; i<=12; i++)
{
        if (indeck[i][4]==4)
                total[3]+=3612; /* 4*43c2 */
        else if (indeck[i][4]==3)
                total[3]+=946; /* 44c2 */
}

total[3]-=total[6];

/* two pair */
for (i=0; i<=11; i++)
{
        for (j=i+1; j<=12; j++)
        {
                if ((indeck[i][4]==2)&&(indeck[j][4]==2))
                        total[2]+=43;

                else if ((indeck[i][4]==2)&&(indeck[j][4]==3))
                        total[2]+=126; /* 3*42 */
                else if ((indeck[i][4]==3)&&(indeck[j][4]==2))
```

```
                        total[2]+=126; /* 3*42 */

                else if ((indeck[i][4]==2)&&(indeck[j][4]==4))
                        total[2]+=246; /* 6*41 */
                else if ((indeck[i][4]==4)&&(indeck[j][4]==2))
                        total[2]+=246; /* 6*41 */

                else if ((indeck[i][4]==3)&&(indeck[j][4]==3))
                        total[2]+=369; /* 3*3*41 */

                else if ((indeck[i][4]==3)&&(indeck[j][4]==4))
                        total[2]+=720; /* 3*6*40 */
                else if ((indeck[i][4]==4)&&(indeck[j][4]==3))
                        total[2]+=720; /* 3*6*40 */

                else if ((indeck[i][4]==4)&&(indeck[j][4]==4))
                        total[2]+=1404; /* 6*6*39 */
        }
}


/* pair */
for (i=9; i<=12; i++)
{
        if (indeck[i][4]==4)
                numcom=6;
        else if (indeck[i][4]==3)
                numcom=3;
        else if (indeck[i][4]==2)
                numcom=1;

        if (indeck[i][4]>=2)
                for (j=0; j<=10; j++)
                        for (k=j+1; k<=11; k++)
                                for (l=k+1; l<=12; l++)
                                        if ((i!=j)&&(i!=k)&&(i!=l))

total[1]+=(numcom*indeck[j][4]*indeck[k][4]*indeck[l][4]);
}


total[0]=850668;
for (i=1; i<=13; i++)
        total[0]-=total[i];
if (total[5]<0)
```

```
                cerr << "total[5]=\t" << total[5] << "\n";
                total[7]+=total[8]+total[9];
                total[8]=total[10];
                total[9]=total[11];
                total[10]=1533939;

}



function fast1(r, s)
{
var i,j,k,l,numcom;

total[0]=0;total[1]=0;total[2]=0;total[3]=0;total[4]=0;total[5]=0;total[6]=0;
total[7]=0;total[8]=0;total[9]=0; total[10]=0; total[11]=0;

if (r>=4)
if (r!=12)
total[8]+=indeck[r-4][s]*indeck[r-3][s]*indeck[r-2][s]*indeck[r-1][s];
else
total[9]=indeck[r-4][s]*indeck[r-3][s]*indeck[r-2][s]*indeck[r-1][s];
if ((r>=3)and(r<=11))
if (r!=11)
total[8]+=indeck[r-3][s]*indeck[r-2][s]*indeck[r-1][s]*indeck[r+1][s];
else
total[9]=indeck[r-3][s]*indeck[r-2][s]*indeck[r-1][s]*indeck[r+1][s];
if ((r>=2)and(r<=10))
if (r!=10)
total[8]+=indeck[r-2][s]*indeck[r-1][s]*indeck[r+1][s]*indeck[r+2][s];
else
total[9]=indeck[r-2][s]*indeck[r-1][s]*indeck[r+1][s]*indeck[r+2][s];
if ((r>=1)and(r<=9))
if (r!=9)
total[8]+=indeck[r-1][s]*indeck[r+1][s]*indeck[r+2][s]*indeck[r+3][s];
else
total[9]=indeck[r-1][s]*indeck[r+1][s]*indeck[r+2][s]*indeck[r+3][s];
if (r<=8)
if (r!=8)
total[8]+=indeck[r+1][s]*indeck[r+2][s]*indeck[r+3][s]*indeck[r+4][s];
else
total[9]=indeck[r+1][s]*indeck[r+2][s]*indeck[r+3][s]*indeck[r+4][s];
```

```
if (r==12)
total[8]+=indeck[0][s]*indeck[1][s]*indeck[2][s]*indeck[3][s];
else if (r==0)
total[8]+=indeck[12][s]*indeck[1][s]*indeck[2][s]*indeck[3][s];
else if (r==1)
total[8]+=indeck[12][s]*indeck[0][s]*indeck[2][s]*indeck[3][s];
else if (r==2)
total[8]+=indeck[12][s]*indeck[0][s]*indeck[1][s]*indeck[3][s];
else if (r==3)
total[8]+=indeck[12][s]*indeck[0][s]*indeck[1][s]*indeck[2][s];


if (r>=4)
total[4]+=indeck[r-4][4]*indeck[r-3][4]*indeck[r-2][4]*indeck[r-1][4];
if ((r>=3)and(r<=11))
total[4]+=indeck[r-3][4]*indeck[r-2][4]*indeck[r-1][4]*indeck[r+1][4];
if ((r>=2)and(r<=10))
total[4]+=indeck[r-2][4]*indeck[r-1][4]*indeck[r+1][4]*indeck[r+2][4];
if ((r>=1)and(r<=9))
total[4]+=indeck[r-1][4]*indeck[r+1][4]*indeck[r+2][4]*indeck[r+3][4];
if (r<=9)
total[4]+=indeck[r+1][4]*indeck[r+2][4]*indeck[r+3][4]*indeck[r+4][4];

if (r==12)
total[4]+=indeck[0][4]*indeck[1][4]*indeck[2][4]*indeck[3][4];
else if (r==0)
total[4]+=indeck[12][4]*indeck[1][4]*indeck[2][4]*indeck[3][4];
else if (r==1)
total[4]+=indeck[12][4]*indeck[0][4]*indeck[2][4]*indeck[3][4];
else if (r==2)
total[4]+=indeck[12][4]*indeck[0][4]*indeck[1][4]*indeck[3][4];
else if (r==3)
total[4]+=indeck[12][4]*indeck[0][4]*indeck[1][4]*indeck[2][4];

total[4]-=(total[8]+total[9]);

if (indeck[13][s]==12)
total[5]=495;
else if (indeck[13][s]==11)
total[5]=330;
else if (indeck[13][s]==10)
total[5]=210;
else if (indeck[13][s]==9)
total[5]=126;
else if (indeck[13][s]==8)
total[5]=70;
```

```
total[5]-=(total[8]+total[9]);

/* four of a kind */
for (i=0; i<=12; i++)
{
if ((i!=r)and(indeck[i][4]==4))
total[7]++;
if ((i!=r)and(indeck[r][4]==3))
total[7]+=indeck[i][4];
}

/* full house */
for (i=0; i<=12; i++)
{
if (i!=r)
{
if (indeck[r][4]==3)
{
if (indeck[i][4]==4)
total[6]+=30;
else if (indeck[i][4]==3)
total[6]+=12;
else if (indeck[i][4]==2)
total[6]+=3;
}
else if (indeck[r][4]==2)
{
if (indeck[i][4]==4)
total[6]+=14;
else if (indeck[i][4]==3)
total[6]+=5;
else if (indeck[i][4]==2)
total[6]++;
}
else if (indeck[r][4]==1)
{
if (indeck[i][4]==4)
total[6]+=4;
else if (indeck[i][4]==3)
total[6]++;
}
}
}

/* three of a kind */
```

```
for (i=0; i<=11; i++)
{
for (j=i+1; j<=12; j++)
{
if ((i!=r)and(j!=r))
{
if (indeck[i][4]==4)
total[3]+=4*indeck[j][4];
else if (indeck[i][4]==3)
total[3]+=indeck[j][4];

if (indeck[j][4]==4)
total[3]+=4*indeck[i][4];
else if (indeck[j][4]==3)
total[3]+=indeck[i][4];

if (indeck[r][4]==3)
total[3]+=(3*indeck[i][4]*indeck[j][4]);
else if (indeck[r][4]==2)
total[3]+=(indeck[i][4]*indeck[j][4]);
}
}
}

/* two pair */
for (i=0; i<=11; i++)
{
for (j=i+1; j<=12; j++)
{
if ((i!=r)and(j!=r))
{
if ((indeck[i][4]==2)and(indeck[j][4]==2))
{
total[2]++;
total[2]+=(indeck[r][4]*4);
}

else if ((indeck[i][4]==2)and(indeck[j][4]==3))
{
total[2]+=3;
total[2]+=(indeck[r][4]*9);
}
else if ((indeck[i][4]==3)and(indeck[j][4]==2))
{
total[2]+=3;
total[2]+=(indeck[r][4]*9);
```

```
}
else if ((indeck[i][4]==2)and(indeck[j][4]==4))
{
total[2]+=6;
total[2]+=(indeck[r][4]*16);
}
else if ((indeck[i][4]==4)and(indeck[j][4]==2))
{
total[2]+=6;
total[2]+=(indeck[r][4]*16);
}
else if ((indeck[i][4]==3)and(indeck[j][4]==3))
{
total[2]+=9;
total[2]+=(indeck[r][4]*18);
}
else if ((indeck[i][4]==3)and(indeck[j][4]==4))
{
total[2]+=18;
total[2]+=(indeck[r][4]*30);
}
else if ((indeck[i][4]==4)and(indeck[j][4]==3))
{
total[2]+=18;
total[2]+=(indeck[r][4]*30);
}
else if ((indeck[i][4]==4)and(indeck[j][4]==4))
{
total[2]+=36;
total[2]+=(indeck[r][4]*48);
}
else if ((indeck[i][4]==1)and(indeck[j][4]==4))
total[2]+=(indeck[r][4]*6);
else if ((indeck[i][4]==1)and(indeck[j][4]==3))
total[2]+=(indeck[r][4]*3);
else if ((indeck[i][4]==1)and(indeck[j][4]==2))
total[2]+=(indeck[r][4]);
else if ((indeck[i][4]==4)and(indeck[j][4]==1))
total[2]+=(indeck[r][4]*6);
else if ((indeck[i][4]==3)and(indeck[j][4]==1))
total[2]+=(indeck[r][4]*3);
else if ((indeck[i][4]==2)and(indeck[j][4]==1))
total[2]+=(indeck[r][4]);
}
}
}
```

```
/* pair */
for (j=0; j<=10; j++)
for (k=j+1; k<=11; k++)
for (l=k+1; l<=12; l++)
if ((r!=j)and(r!=k)and(r!=l))
{
if (r>=9)
total[1]+=(indeck[r][4]*indeck[j][4]*indeck[k][4]*indeck[l][4]);

if (j>=9)
{
if (indeck[j][4]==4)
numcom=6;
else if (indeck[j][4]==3)
numcom=3;
else if (indeck[j][4]==2)
numcom=1;
else
numcom=0;
total[1]+=(numcom*indeck[k][4]*indeck[l][4]);
}

if (k>=9)
{
if (indeck[k][4]==4)
numcom=6;
else if (indeck[k][4]==3)
numcom=3;
else if (indeck[k][4]==2)
numcom=1;
else
numcom=0;
total[1]+=(numcom*indeck[j][4]*indeck[l][4]);
}

if (l>=9)
{
if (indeck[l][4]==4)
numcom=6;
else if (indeck[l][4]==3)
numcom=3;
else if (indeck[l][4]==2)
numcom=1;
else
numcom=0;
```

```
total[1]+=(numcom*indeck[j][4]*indeck[k][4]);
}}

total[0]=178365;
for (i=1; i<=9; i++)total[0]-=total[i];

                total[10]=178365;

}



function fast2(r1,s1,r2,s2)
{
var i,j,k,hold;
total[0]=0;total[1]=0;total[2]=0;total[3]=0;total[4]=0;total[5]=0;total[6]=0;
total[7]=0;total[8]=0;total[9]=0;total[10]=0; total[11]=0;

        if (r2<r1)
        {
                hold=r1;
                r1=r2;
                r2=hold;
                hold=s1;
                s1=s2;
                s2=hold;
        }

        /* straight and royal flush */
        if ((r1+1==r2)&&(s1==s2))
        {
                if (r1>=3)
                        if (r2!=12)
                                total[10]+=indeck[r1-3][s1]*indeck[r1-2][s1]*indeck[r1-
1][s1];
                        else
                                total[11]=indeck[r1-3][s1]*indeck[r1-2][s1]*indeck[r1-
1][s1];
                if ((r1>=2)&&(r2<=11))
                        if (r2!=11)
                                total[10]+=indeck[r1-2][s1]*indeck[r1-
1][s1]*indeck[r2+1][s1];
                        else
                                total[11]=indeck[r1-2][s1]*indeck[r1-
1][s1]*indeck[r2+1][s1];
                if ((r1>=1)&&(r2<=10))
```

```
                    if (r2!=10)
                            total[10]+=indeck[r1-
1][s1]*indeck[r2+1][s1]*indeck[r2+2][s1];
                    else
                            total[11]=indeck[r1-
1][s1]*indeck[r2+1][s1]*indeck[r2+2][s1];
            if (r2<=9)
                    if (r2!=9)

        total[10]+=indeck[r2+1][s1]*indeck[r2+2][s1]*indeck[r2+3][s1];
                    else

        total[11]=indeck[r2+1][s1]*indeck[r2+2][s1]*indeck[r2+3][s1];
        }
        else if ((r1+2==r2)&&(s1==s2))
        {
                if (r1>=2)
                        if (r2!=12)
                                total[10]+=indeck[r1-2][s1]*indeck[r1-
1][s1]*indeck[r1+1][s1];
                        else // (10)-(j)-q-(k)-a
                                total[11]=indeck[8][s1]*indeck[9][s1]*indeck[11][s1];
                if ((r1>=1)&&(r2<=11))
                        if (r2!=11)
                                total[10]+=indeck[r1-
1][s1]*indeck[r1+1][s1]*indeck[r2+1][s1];
                        else // (10)-j-(q)-k-(a)
                                total[11]=indeck[8][s1]*indeck[10][s1]*indeck[12][s1];
                if (r2<=10)
                        if (r2!=10)

        total[10]+=indeck[r1+1][s1]*indeck[r2+1][s1]*indeck[r2+2][s1];
                        else // 10-(j)-q-(k)-(a)
                                total[11]=indeck[9][s1]*indeck[11][s1]*indeck[12][s1];
        }
        else if ((r1+3==r2)&&(s1==s2))
        {
                 if (r1>=1)
                        if (r2!=12)
                                total[10]+=indeck[r1-
 1][s1]*indeck[r1+1][s1]*indeck[r1+2][s1];
                        else
                                total[11]=indeck[r1-
1][s1]*indeck[r1+1][s1]*indeck[r1+2][s1];
                 if (r2<=11)
                        if (r2!=11)
```

```
total[10]+=indeck[r1+1][s1]*indeck[r1+2][s1]*indeck[r2+1][s1];
                else

total[11]=indeck[r1+1][s1]*indeck[r1+2][s1]*indeck[r2+1][s1];
}
else if ((r1+4==r2)&&(s1==s2))
{
        if (r2!=12)
                total[10]+=indeck[r1+1][s1]*indeck[r1+2][s1]*indeck[r1+3][s1];
        else
                total[11]=indeck[r1+1][s1]*indeck[r1+2][s1]*indeck[r1+3][s1];
}

if ((r1==0)&&(r2==12)&&(s1==s2))
        total[10]+=indeck[1][s1]*indeck[2][s1]*indeck[3][s1];
else if ((r1==1)&&(r2==12)&&(s1==s2))
        total[10]+=indeck[0][s1]*indeck[2][s1]*indeck[3][s1];
else if ((r1==2)&&(r2==12)&&(s1==s2))
        total[10]+=indeck[0][s1]*indeck[1][s1]*indeck[3][s1];
else if ((r1==3)&&(r2==12)&&(s1==s2))
        total[10]+=indeck[0][s1]*indeck[1][s1]*indeck[2][s1];
else if ((r1==0)&&(r2==1)&&(s1==s2))
        total[10]+=indeck[2][s1]*indeck[3][s1]*indeck[12][s1];
else if ((r1==0)&&(r2==2)&&(s1==s2))
        total[10]+=indeck[1][s1]*indeck[3][s1]*indeck[12][s1];
else if ((r1==0)&&(r2==3)&&(s1==s2))
        total[10]+=indeck[1][s1]*indeck[2][s1]*indeck[12][s1];
else if ((r1==1)&&(r2==2)&&(s1==s2))
        total[10]+=indeck[0][s1]*indeck[3][s1]*indeck[12][s1];
else if ((r1==1)&&(r2==3)&&(s1==s2))
        total[10]+=indeck[0][s1]*indeck[2][s1]*indeck[12][s1];
else if ((r1==2)&&(r2==3)&&(s1==s2))
        total[10]+=indeck[0][s1]*indeck[1][s1]*indeck[12][s1];


/* straight */

if (r1+1==r2)
{
        if (r1>=3)
                total[4]+=indeck[r1-3][4]*indeck[r1-2][4]*indeck[r1-1][4];
        if ((r1>=2)&&(r2<=11))
                total[4]+=indeck[r1-2][4]*indeck[r1-1][4]*indeck[r2+1][4];
        if ((r1>=1)&&(r2<=10))
                total[4]+=indeck[r1-1][4]*indeck[r2+1][4]*indeck[r2+2][4];
```

```
        if (r2<=9)
                total[4]+=indeck[r2+1][4]*indeck[r2+2][4]*indeck[r2+3][4];
}
else if (r1+2==r2)
{
        if (r1>=2)
                total[4]+=indeck[r1-2][4]*indeck[r1-1][4]*indeck[r1+1][4];
        if ((r1>=1)&&(r2<=11))
                total[4]+=indeck[r1-1][4]*indeck[r1+1][4]*indeck[r2+1][4];
        if (r2<=10)
                total[4]+=indeck[r1+1][4]*indeck[r2+1][4]*indeck[r2+2][4];
}
else if (r1+3==r2)
{
        if (r1>=1)
                total[4]+=indeck[r1-1][4]*indeck[r1+1][4]*indeck[r1+2][4];
        if (r2<=11)
                total[4]+=indeck[r1+1][4]*indeck[r1+2][4]*indeck[r2+1][4];
}
else if (r1+4==r2)
{
        total[4]+=indeck[r1+1][4]*indeck[r1+2][4]*indeck[r1+3][4];
}

if ((r1==0)&&(r2==12))
        total[4]+=indeck[1][4]*indeck[2][4]*indeck[3][4];
else if ((r1==1)&&(r2==12))
        total[4]+=indeck[0][4]*indeck[2][4]*indeck[3][4];
else if ((r1==2)&&(r2==12))
        total[4]+=indeck[0][4]*indeck[1][4]*indeck[3][4];
else if ((r1==3)&&(r2==12))
        total[4]+=indeck[0][4]*indeck[1][4]*indeck[2][4];
else if ((r1==0)&&(r2==1))
        total[4]+=indeck[2][4]*indeck[3][4]*indeck[12][4];
else if ((r1==0)&&(r2==2))
        total[4]+=indeck[1][4]*indeck[3][4]*indeck[12][4];
else if ((r1==0)&&(r2==3))
        total[4]+=indeck[1][4]*indeck[2][4]*indeck[12][4];
else if ((r1==1)&&(r2==2))
        total[4]+=indeck[0][4]*indeck[3][4]*indeck[12][4];
else if ((r1==1)&&(r2==3))
        total[4]+=indeck[0][4]*indeck[2][4]*indeck[12][4];
else if ((r1==2)&&(r2==3))
        total[4]+=indeck[0][4]*indeck[1][4]*indeck[12][4];
```

```
total[4]-=(total[10]+total[11]);

if (s1==s2)
{
        if (indeck[13][s1]==11)
                total[5]=165;
        else if (indeck[13][s1]==10)
                total[5]=120;
        else if (indeck[13][s1]==9)
                total[5]=84;
        else if (indeck[13][s1]==8)
                total[5]=56;
        else if (indeck[13][s1]==7)
                total[5]=35;
        else if (indeck[13][s1]==6)
                total[5]=20;
        else if (indeck[13][s1]==5)
                total[5]=10;
        else if (indeck[13][s1]==4)
                total[5]=4;
        else if (indeck[13][s1]==3)
                total[5]=1;
}


total[5]-=(total[10]+total[11]);


/* four of a kind */
for (i=0; i<=12; i++)
{
        if ((r1==r2)&&(i!=r1)&&(indeck[r1][4]==2))
        {
                if (r1==12)
                {
                        if (i<=2)
                                total[9]+=indeck[i][4];
                        else
                                total[9]+=indeck[i][4];
                }
                else if (r1<=2)
                {
                        if ((i<=2)||(i==12))
                                total[8]+=indeck[i][4];
                        else
                                total[8]+=indeck[i][4];
```

```
                }
                else
                        total[7]+=indeck[i][4];
        }
        else if ((r1!=r2)&&(i==r1)&&(indeck[r1][4]==3))
        {
                if (r1==12)
                {
                        if (r2<=2)
                                total[9]++;
                        else
                                total[9]++;
                }
                else if (r1<=2)
                {
                        if ((r2<=2)||(r2==12))
                                total[8]++;
                        else
                                total[8]++;
                }
                else
                        total[7]++;
        }
        else if ((r1!=r2)&&(i==r2)&&(indeck[r2][4]==3))
        {
                if (r2==12)
                {
                        if (r1<=2)
                                total[9]++;
                        else
                                total[9]++;
                }
                else if (r2<=2)
                {
                        if ((r1<=2)||(r1==12))
                                total[8]++;
                        else
                                total[8]++;
                }
                else
                        total[7]++;
        }
}

/* full house */
if (r1==r2)
```

```
{
        for (i=0; i<=12; i++)
        {
                if (i!=r1)
                {
                        if (indeck[i][4]==4)
                        {
                                total[6]+=4;
                                if (indeck[r1][4]==2)
                                        total[6]+=12;
                                else if (indeck[r1][4]==1)
                                        total[6]+=6;
                        }
                        else if (indeck[i][4]==3)
                        {
                                total[6]++;
                                if (indeck[r1][4]==2)
                                        total[6]+=6;
                                else if (indeck[r1][4]==1)
                                        total[6]+=3;
                        }
                        else if (indeck[i][4]==2)
                        {
                                if (indeck[r1][4]==2)
                                        total[6]+=2;
                                else if (indeck[r1][4]==1)
                                        total[6]++;
                        }
                }
        }
}
else
{
        if ((indeck[r1][4]==3)&&(indeck[r2][4]==3))
                total[6]=18;
        else if ((indeck[r1][4]==3)&&(indeck[r2][4]==2))
                total[6]=9;
        else if ((indeck[r1][4]==2)&&(indeck[r2][4]==3))
                total[6]=9;
        else if ((indeck[r1][4]==3)&&(indeck[r2][4]==1))
                total[6]=3;
        else if ((indeck[r1][4]==1)&&(indeck[r2][4]==3))
                total[6]=3;
        else if ((indeck[r1][4]==2)&&(indeck[r2][4]==2))
                total[6]=4;
        else if ((indeck[r1][4]==2)&&(indeck[r2][4]==1))
```

```
                total[6]=1;
        else if ((indeck[r1][4]==1)&&(indeck[r2][4]==2))
                total[6]=1;
}



/* three of a kind */
if (r1!=r2)
{
        for (i=0; i<=12; i++)
        {
                if ((i!=r1)&&(i!=r2))
                {
                        if (indeck[i][4]==4)
                                total[3]+=4;
                        else if (indeck[i][4]==3)
                                total[3]++;

                        if (indeck[r1][4]==3)
                                total[3]+=3*indeck[i][4];
                        else if (indeck[r1][4]==2)
                                total[3]+=indeck[i][4];

                        if (indeck[r2][4]==3)
                                total[3]+=3*indeck[i][4];
                        else if (indeck[r2][4]==2)
                                total[3]+=indeck[i][4];
                }
        }
}
else
        for (i=0; i<=11; i++)
                for (j=i+1; j<=12; j++)
                        if ((i!=r1)&&(j!=r1))
                                total[3]+=indeck[r1][4]*indeck[i][4]*indeck[j][4];



/* two pair */
if (r1!=r2)
{
        for (i=0; i<=12; i++)
        {
                if ((i!=r1)&&(i!=r2))
```

```
                {
                        total[2]+=indeck[i][4]*indeck[r1][4]*indeck[r2][4];

                        if (indeck[i][4]==4)
                                total[2]+=6*(indeck[r1][4]+indeck[r2][4]);
                        else if (indeck[i][4]==3)
                                total[2]+=3*(indeck[r1][4]+indeck[r2][4]);
                        if (indeck[i][4]==2)
                                total[2]+=(indeck[r1][4]+indeck[r2][4]);
                }
        }
}
else
        for (i=0; i<=11; i++)
                for (j=i+1; j<=12; j++)
                        if ((i!=r1)&&(j!=r1))
                        {
                                if (indeck[i][4]==4)
                                        total[2]+=(6*indeck[j][4]);
                                else if (indeck[i][4]==3)
                                        total[2]+=(3*indeck[j][4]);
                                else if (indeck[i][4]==2)
                                        total[2]+=indeck[j][4];

                                if (indeck[j][4]==4)
                                        total[2]+=(6*indeck[i][4]);
                                else if (indeck[j][4]==3)
                                        total[2]+=(3*indeck[i][4]);
                                else if (indeck[j][4]==2)
                                        total[2]+=indeck[i][4];
                        }


/* high pair */
if ((r1!=r2)&&(r1>=9)&&(r2>=9))
{
        for (i=0; i<=11; i++)
        {
                for (j=i+1; j<=12; j++)
                {
                        if ((i!=r1)&&(i!=r2)&&(j!=r1)&&(j!=r2))
                        {

total[1]+=((indeck[r1][4]+indeck[r2][4])*indeck[i][4]*indeck[j][4]);
```

```
                        if (i>=9)
                                if (indeck[i][4]==4)
                                        total[1]+=(6*indeck[j][4]);
                                else if (indeck[i][4]==3)
                                        total[1]+=(3*indeck[j][4]);
                                else if (indeck[i][4]==2)
                                        total[1]+=indeck[j][4];

                        if (j>=9)
                                if (indeck[j][4]==4)
                                        total[1]+=(6*indeck[i][4]);
                                else if (indeck[j][4]==3)
                                        total[1]+=(3*indeck[i][4]);
                                else if (indeck[j][4]==2)
                                        total[1]+=indeck[i][4];
                        }
                }
        }
}

else if ((r1!=r2)&&(r1>=9)&&(r2<=8))
{
        for (i=0; i<=11; i++)
        {
                for (j=i+1; j<=12; j++)
                {
                        if ((i!=r1)&&(i!=r2)&&(j!=r1)&&(j!=r2))
                        {
                                total[1]+=(indeck[r1][4]*indeck[i][4]*indeck[j][4]);

                                if (i>=9)
                                        if (indeck[i][4]==4)
                                                total[1]+=(6*indeck[j][4]);
                                        else if (indeck[i][4]==3)
                                                total[1]+=(3*indeck[j][4]);
                                        else if (indeck[i][4]==2)
                                                total[1]+=indeck[j][4];

                                if (j>=9)
                                        if (indeck[j][4]==4)
                                                total[1]+=(6*indeck[i][4]);
                                        else if (indeck[j][4]==3)
                                                total[1]+=(3*indeck[i][4]);
                                        else if (indeck[j][4]==2)
                                                total[1]+=indeck[i][4];
                        }
```

```
                }
        }
}


else if ((r1!=r2)&&(r1<=8)&&(r2>=9))
{
        for (i=0; i<=11; i++)
        {
                for (j=i+1; j<=12; j++)
                {
                        if ((i!=r1)&&(i!=r2)&&(j!=r1)&&(j!=r2))
                        {
                                total[1]+=(indeck[r2][4]*indeck[i][4]*indeck[j][4]);

                                if (i>=9)
                                {
                                        if (indeck[i][4]==4)
                                                total[1]+=(6*indeck[j][4]);
                                        else if (indeck[i][4]==3)
                                                total[1]+=(3*indeck[j][4]);
                                        else if (indeck[i][4]==2)
                                                total[1]+=indeck[j][4];
                                }
                                if (j>=9)
                                {
                                        if (indeck[j][4]==4)
                                                total[1]+=(6*indeck[i][4]);
                                        else if (indeck[j][4]==3)
                                                total[1]+=(3*indeck[i][4]);
                                        else if (indeck[j][4]==2)
                                                total[1]+=indeck[i][4];
                                }
                        }
                }
        }
}


else if ((r1!=r2)&&(r1<=8)&&(r2<=8))
{
        for (i=0; i<=11; i++)
        {
                for (j=i+1; j<=12; j++)
                {
                        if ((i!=r1)&&(i!=r2)&&(j!=r1)&&(j!=r2))
```

```
                        {
                                if (i>=9)
                                {
                                        if (indeck[i][4]==4)
                                                total[1]+=(6*indeck[j][4]);
                                        else if (indeck[i][4]==3)
                                                total[1]+=(3*indeck[j][4]);
                                        else if (indeck[i][4]==2)
                                                total[1]+=indeck[j][4];
                                }
                                if (j>=9)
                                {
                                        if (indeck[j][4]==4)
                                                total[1]+=(6*indeck[i][4]);
                                        else if (indeck[j][4]==3)
                                                total[1]+=(3*indeck[i][4]);
                                        else if (indeck[j][4]==2)
                                                total[1]+=indeck[i][4];
                                }
                        }
                }
        }
}


else if ((r1==r2)&&(r1>=9))
{
        for (i=0; i<=10; i++)
        {
                for (j=i+1; j<=11; j++)
                {
                        for (k=j+1; k<=12; k++)
                        {
                                if ((i!=r1)&&(j!=r1)&&(k!=r1))

total[1]+=(indeck[i][4]*indeck[j][4]*indeck[k][4]);
                        }
                }
        }
}

total[0]=11480;
for (i=1; i<=13; i++)
        total[0]-=total[i];

        total[7]+=total[8]+total[9];
```

```
                total[8]=total[10];
                total[9]=total[11];
                total[10]=16215;

}



function fast3(r1,s1,r2,s2,r3,s3)
{
var i,j,hold,flush;

        total[0]=0;total[1]=0;total[2]=0;total[3]=0;total[4]=0;total[5]=0;total[6]=0;
        total[7]=0;total[8]=0;total[9]=0; total[10]=0; total[11]=0;

        if (r2>r3) {hold=r2; r2=r3; r3=hold; hold=s2; s2=s3; s3=hold;}
        if (r1>r2) {hold=r2; r2=r1; r1=hold; hold=s2; s2=s1; s1=hold;}
        if (r2>r3) {hold=r2; r2=r3; r3=hold; hold=s2; s2=s3; s3=hold;}



        if ((s1==s2)&&(s2==s3))
                flush=1;
        else
                flush=0;



        if (flush==1)
        {
                /* royal flush */
                if ((r1==8)&&(r2==9)&&(r3==10))
                        total[11]=indeck[11][s1]*indeck[12][s1];
                else if ((r1==8)&&(r2==9)&&(r3==11))
                        total[11]=indeck[10][s1]*indeck[12][s1];
                else if ((r1==8)&&(r2==9)&&(r3==12))
                        total[11]=indeck[10][s1]*indeck[11][s1];
                else if ((r1==8)&&(r2==10)&&(r3==11))
                        total[11]=indeck[9][s1]*indeck[12][s1];
                else if ((r1==8)&&(r2==10)&&(r3==12))
                        total[11]=indeck[9][s1]*indeck[11][s1];
                else if ((r1==8)&&(r2==11)&&(r3==12))
                        total[11]=indeck[9][s1]*indeck[10][s1];
                else if ((r1==9)&&(r2==10)&&(r3==11))
                        total[11]=indeck[8][s1]*indeck[12][s1];
```

```
else if ((r1==9)&&(r2==10)&&(r3==12))
        total[11]=indeck[8][s1]*indeck[11][s1];
else if ((r1==9)&&(r2==11)&&(r3==12))
        total[11]=indeck[8][s1]*indeck[10][s1];
else if ((r1==10)&&(r2==11)&&(r3==12))
        total[11]=indeck[8][s1]*indeck[9][s1];

/* straight flush */
if ((r2==r1+1)&&(r3==r1+2))
{
        if (r1>=2)
                total[10]=indeck[r1-2][s1]*indeck[r1-1][s1];
        if ((r1>=1)&&(r3<=11))
                total[10]+=indeck[r1-1][s1]*indeck[r3+1][s1];
        if (r3<=10)
                total[10]+=indeck[r3+1][s1]*indeck[r3+2][s1];
}
else if ((r2==r1+2)&&(r3==r1+3))
{
        if (r1>=1)
                total[10]=indeck[r1-1][s1]*indeck[r1+1][s1];
        if (r3<=11)
                total[10]+=indeck[r1+1][s1]*indeck[r3+1][s1];
}
else if ((r2==r1+1)&&(r3==r1+3))
{
        if (r1>=1)
                total[10]=indeck[r1-1][s1]*indeck[r1+2][s1];
        if (r3<=11)
                total[10]+=indeck[r1+2][s1]*indeck[r3+1][s1];
}
else if ((r2==r1+1)&&(r3==r1+4))
        total[10]=indeck[r1+2][s1]*indeck[r1+3][s1];
else if ((r2==r1+2)&&(r3==r1+4))
        total[10]=indeck[r1+1][s1]*indeck[r1+3][s1];
else if ((r2==r1+3)&&(r3==r1+4))
        total[10]=indeck[r1+1][s1]*indeck[r1+2][s1];


if ((r1==0)&&(r2==1)&&(r3==2))
        total[10]+=indeck[3][s1]*indeck[12][s1];
else if ((r1==0)&&(r2==1)&&(r3==3))
        total[10]+=indeck[2][s1]*indeck[12][s1];
else if ((r1==0)&&(r2==2)&&(r3==3))
        total[10]+=indeck[1][s1]*indeck[12][s1];
else if ((r1==1)&&(r2==2)&&(r3==3))
```

```
                total[10]+=indeck[0][s1]*indeck[12][s1];
        else if ((r1==0)&&(r2==1)&&(r3==12))
                total[10]+=indeck[2][s1]*indeck[3][s1];
        else if ((r1==0)&&(r2==2)&&(r3==12))
                total[10]+=indeck[1][s1]*indeck[3][s1];
        else if ((r1==0)&&(r2==3)&&(r3==12))
                total[10]+=indeck[1][s1]*indeck[2][s1];
        else if ((r1==1)&&(r2==2)&&(r3==12))
                total[10]+=indeck[0][s1]*indeck[3][s1];
        else if ((r1==1)&&(r2==3)&&(r3==12))
                total[10]+=indeck[0][s1]*indeck[2][s1];
        else if ((r1==2)&&(r2==3)&&(r3==12))
                total[10]+=indeck[0][s1]*indeck[1][s1];

        total[10]-=total[11];
}


/* straight */

if ((r2==r1+1)&&(r3==r1+2))
{
        if (r1>=2)
                total[4]=indeck[r1-2][4]*indeck[r1-1][4];
        if ((r1>=1)&&(r3<=11))
                total[4]+=(indeck[r1-1][4]*indeck[r3+1][4]);
        if (r3<=10)
                total[4]+=(indeck[r3+1][4]*indeck[r3+2][4]);
}
else if ((r2==r1+2)&&(r3==r1+3))
{
        if (r1>=1)
                total[4]=indeck[r1-1][4]*indeck[r1+1][4];
        if (r3<=11)
                total[4]+=indeck[r1+1][4]*indeck[r3+1][4];
}
else if ((r2==r1+1)&&(r3==r1+3))
{
        if (r1>=1)
                total[4]=indeck[r1-1][4]*indeck[r1+2][4];
        if (r3<=11)
                total[4]+=indeck[r1+2][4]*indeck[r3+1][4];
}
else if ((r2==r1+1)&&(r3==r1+4))
        total[4]=indeck[r1+2][4]*indeck[r1+3][4];
else if ((r2==r1+2)&&(r3==r1+4))
```

```
        total[4]=indeck[r1+1][4]*indeck[r1+3][4];
else if ((r2==r1+3)&&(r3==r1+4))
        total[4]=indeck[r1+1][4]*indeck[r1+2][4];

if ((r1==0)&&(r2==1)&&(r3==2))
        total[4]+=indeck[3][4]*indeck[12][4];
else if ((r1==0)&&(r2==1)&&(r3==3))
        total[4]+=indeck[2][4]*indeck[12][4];
else if ((r1==0)&&(r2==2)&&(r3==3))
        total[4]+=indeck[1][4]*indeck[12][4];
else if ((r1==1)&&(r2==2)&&(r3==3))
        total[4]+=indeck[0][4]*indeck[12][4];
else if ((r1==0)&&(r2==1)&&(r3==12))
        total[4]+=indeck[2][4]*indeck[3][4];
else if ((r1==0)&&(r2==2)&&(r3==12))
        total[4]+=indeck[1][4]*indeck[3][4];
else if ((r1==0)&&(r2==3)&&(r3==12))
        total[4]+=indeck[1][4]*indeck[2][4];
else if ((r1==1)&&(r2==2)&&(r3==12))
        total[4]+=indeck[0][4]*indeck[3][4];
else if ((r1==1)&&(r2==3)&&(r3==12))
        total[4]+=indeck[0][4]*indeck[2][4];
else if ((r1==2)&&(r2==3)&&(r3==12))
        total[4]+=indeck[0][4]*indeck[1][4];

total[4]-=(total[10]+total[11]);

if (flush==1)
{
        if (indeck[13][s1]==10)
                total[5]=45;
        else if (indeck[13][s1]==9)
                total[5]=36;
        else if (indeck[13][s1]==8)
                total[5]=28;
        else if (indeck[13][s1]==7)
                total[5]=21;
        else if (indeck[13][s1]==6)
                total[5]=15;
        else if (indeck[13][s1]==5)
                total[5]=10;
        else if (indeck[13][s1]==4)
                total[5]=6;
        else if (indeck[13][s1]==3)
                total[5]=3;
        else if (indeck[13][s1]==2)
```

```
                total[5]=1;
}

total[5]-=(total[10]+total[11]);


// four of a kind
if ((r1==r3)&&(indeck[r1][4]==1))
{
        if (r1==12)
        {
                for (i=0; i<=11; i++)
                        total[9]+=indeck[i][4];
        }
        else if (r1<=2)
        {
                for (i=0; i<=12; i++)
                        if (i!=r1)
                                total[8]+=indeck[i][4];
        }
        else
        {
                for (i=0; i<=12; i++)
                        if (i!=r1)
                                total[7]+=indeck[i][4];
        }
}
else if ((r1==r2)&&(indeck[r1][4]==2))
{
        if (r1==12)
        {
                total[9]=1;
        }
        else if (r1<=2)
        {
                total[8]=1;
        }
        else
                total[7]=1;
}
else if ((r2==r3)&&(indeck[r2][4]==2))
{
        if (r2==12)
                total[9]=1;
        else if (r2<=2)
                total[8]=1;
```

```
        else
                total[7]=1;
}

/* full house */
if (r1==r3)
{
        for (i=0; i<=12; i++)
        {
                if (i!=r1)
                {
                        if (indeck[i][4]==4)
                                total[6]+=6;
                        else if (indeck[i][4]==3)
                                total[6]+=3;
                        else if (indeck[i][4]==2)
                                total[6]++;
                }
        }
}
else if (r1==r2)
{
        if (indeck[r3][4]==3)
                total[6]+=3;
        else if (indeck[r3][4]==2)
                total[6]++;
        total[6]+=(indeck[r1][4]*indeck[r3][4]);
}
else if (r2==r3)
{
        if (indeck[r1][4]==3)
                total[6]+=3;
        else if (indeck[r1][4]==2)
                total[6]++;
        total[6]+=(indeck[r1][4]*indeck[r3][4]);
}



/* three of a kind */
if (r1==r3)
{
        for (i=0; i<=11; i++)
                for (j=i+1; j<=12; j++)
                        if ((i!=r1)&&(j!=r1))
                                total[3]+=(indeck[i][4]*indeck[j][4]);
```

```
}
else if (r1==r2)
{
        for (i=0; i<=12; i++)
                if ((i!=r1)&&(i!=r3))
                        total[3]+=(indeck[r1][4]*indeck[i][4]);
}
else if (r2==r3)
{
        for (i=0; i<=12; i++)
                if ((i!=r1)&&(i!=r2))
                        total[3]+=(indeck[r2][4]*indeck[i][4]);
}
else
{
        if (indeck[r1][4]==3)
                total[3]+=3;
        else if (indeck[r1][4]==2)
                total[3]++;
        if (indeck[r2][4]==3)
                total[3]+=3;
        else if (indeck[r2][4]==2)
                total[3]++;
        if (indeck[r3][4]==3)
                total[3]+=3;
        else if (indeck[r3][4]==2)
                total[3]++;
}

/* two pair */
if ((r1==r2)&&(r2!=r3))
{
        for (i=0; i<=12; i++)
        {
                if ((i!=r1)&&(i!=r3))
                {
                        total[2]+=(indeck[r3][4]*indeck[i][4]);
                        if (indeck[i][4]==4)
                                total[2]+=6;
                        else if (indeck[i][4]==3)
                                total[2]+=3;
                        else if (indeck[i][4]==2)
                                total[2]++;
                }
        }
}
```

```
else if ((r1!=r2)&&(r2==r3))
{
        for (i=0; i<=12; i++)
        {
                if ((i!=r1)&&(i!=r3))
                {
                        total[2]+=(indeck[r1][4]*indeck[i][4]);
                        if (indeck[i][4]==4)
                                total[2]+=6;
                        else if (indeck[i][4]==3)
                                total[2]+=3;
                        else if (indeck[i][4]==2)
                                total[2]++;
                }
        }
}
else if ((r1!=r2)&&(r2!=r3))
{
        total[2]+=(indeck[r1][4]*indeck[r2][4]);
        total[2]+=(indeck[r1][4]*indeck[r3][4]);
        total[2]+=(indeck[r2][4]*indeck[r3][4]);
}



/* high pair */
if ((r3<=8)&&(r2>r1)&&(r3>r2)) /* three different low cards */
{
        for (i=9; i<=12; i++)
        {
                if (indeck[i][4]==4)
                        total[1]+=6;
                else if (indeck[i][4]==3)
                        total[1]+=3;
                else if (indeck[i][4]==2)
                        total[1]++;
        }
}
else if ((r2>=9)&&(r2>r1)&&(r2==r3)) /* high pair, one lower singleton */
{
        for (i=0; i<=11; i++)
        {
                for (j=i+1; j<=12; j++)
                {
                        if ((i!=r1)&&(i!=r2)&&(j!=r1)&&(j!=r2))
                        {
```

```
                                total[1]+=(indeck[i][4]*indeck[j][4]);
                        }
                }
        }
}
else if ((r2>=9)&&(r2<r3)&&(r2==r1)) /* high pair, one higher singleton */
{
        for (i=0; i<=11; i++)
        {
                for (j=i+1; j<=12; j++)
                {
                        if ((i!=r3)&&(i!=r2)&&(j!=r3)&&(j!=r2))
                        {
                                total[1]+=(indeck[i][4]*indeck[j][4]);
                        }
                }
        }
}
else if ((r2<=8)&&(r3>=9)&&(r2>r1)) /* one high card, two low */
{
        for (i=0; i<=12; i++)
        {
                if ((i!=r1)&&(i!=r2)&&(i!=r3))
                {
                        total[1]+=(indeck[i][4]*indeck[r3][4]);
                }
        }
        for (i=9; i<=12; i++)
        {
                if (i!=r3)
                {
                        if (indeck[i][4]==4)
                                total[1]+=6;
                        else if (indeck[i][4]==3)
                                total[1]+=3;
                        else if (indeck[i][4]==2)
                                total[1]++;
                }
        }
}
else if ((r1<=8)&&(r2>=9)&&(r3>r2)) /* two high cards, one low */
{
        for (i=0; i<=12; i++)
        {
                if ((i!=r1)&&(i!=r2)&&(i!=r3))
                {
```

```
                        total[1]+=(indeck[i][4]*indeck[r2][4]);
                        total[1]+=(indeck[i][4]*indeck[r3][4]);
                }
        }
        for (i=9; i<=12; i++)
        {
                if ((i!=r2)&&(i!=r3))
                {
                        if (indeck[i][4]==4)
                                total[1]+=6;
                        else if (indeck[i][4]==3)
                                total[1]+=3;
                        else if (indeck[i][4]==2)
                                total[1]++;
                }
        }
}
else if ((r1>=9)&&(r2>r1)&&(r3>r2)) /* three high cards */
{
        for (i=0; i<=12; i++)
        {
                if ((i!=r1)&&(i!=r2)&&(i!=r3))
                {
                        total[1]+=(indeck[i][4]*indeck[r1][4]);
                        total[1]+=(indeck[i][4]*indeck[r2][4]);
                        total[1]+=(indeck[i][4]*indeck[r3][4]);
                }
        }
        for (i=9; i<=12; i++)
        {
                if ((i!=r1)&&(i!=r2)&&(i!=r3))
                {
                        if (indeck[i][4]==4)
                                total[1]+=6;
                        else if (indeck[i][4]==3)
                                total[1]+=3;
                        else if (indeck[i][4]==2)
                                total[1]++;
                }
        }
}

total[0]=861;
for (i=1; i<=13; i++)
        total[0]-=total[i];
```

```
                total[7]+=total[8]+total[9];
                total[8]=total[10];
                total[9]=total[11];
                total[10]=1081;

}


function slow4(c1,c2,c3,c4)
{
        total=[];
        total[0]=0;total[1]=0;total[2]=0;total[3]=0;total[4]=0;total[5]=0;total[6]=0;
        total[7]=0;total[8]=0;total[9]=0; total[10]=0; total[11]=0;

        for (c5=1;c5<53;++c5) {
                if ((c5!=card[0]) and (c5!=card[1]) and (c5!=card[2]) and (c5!=card[3])
                        and (c5!=card[4])) {
                                payhand(c1,c2,c3,c4,c5);
                                rank=10-rank;
                                total[rank]++;
                }
        }
        total[10]=47;
}



function convert(n)
{
        s=0; r=n;
        if (r>13) {r=r-13; s++;}
        if (r>13) {r=r-13; s++;}
        if (r>13) {r=r-13; s++;}
        r=r-2;
        if (r==-1) {r=12;}
}

function calculate(a,b,c,d,e)
{
total=[];
setinitialdeck();
convert (a); ar=r; as=s; indeck [r][s]=0; indeck[13][s]--; indeck[r][4]--;
convert (b); br=r; bs=s; indeck [r][s]=0; indeck[13][s]--; indeck[r][4]--;
convert (c); cr=r; cs=s; indeck [r][s]=0; indeck[13][s]--; indeck[r][4]--;
convert (d); dr=r; ds=s; indeck [r][s]=0; indeck[13][s]--; indeck[r][4]--;
```

```
convert (e); er=r; es=s; indeck [r][s]=0; indeck[13][s]--; indeck[r][4]--;

}

function findbottom () {
        pay[0]=0;
        for (next=bott-1; next>0; --next) {
                if ((pay[next]!=0) or (next==0)) {return;}
        }
}

function findhighest () {
        highest=bott;
        for (y=bott+1; y<10; y++) {
                if (pay[y]>pay[highest]) {highest=y;}
        }
}



function adjusttable2() {
x=0; holdrank=0; holdjs=0; hold3=0; hold2=0;
// holdjs is 1 when player holds 3 cards including Js or better
//hold3 is 1 when player holds 4 cards with 3oak
//hold is 1 when player holds 4 cards with high pair
// x is # of cards held
// first we need to call the appropriate fast or slow routine

x=hold[0]+hold[1]+hold[2]+hold[3]+hold[4];
if (x==0) {calculate (card[0],card[1],card[2],card[3],card[4]);
                fast0();
                }


if (x==1) {
                        if (hold[0]==1) {calculate
(card[0],card[1],card[2],card[3],card[4]);}
                        if (hold[1]==1) {calculate
(card[1],card[0],card[2],card[3],card[4]);}
                        if (hold[2]==1) {calculate
(card[2],card[1],card[0],card[3],card[4]);}
                        if (hold[3]==1) {calculate
(card[3],card[1],card[2],card[0],card[4]);}
                        if (hold[4]==1) {calculate
(card[4],card[1],card[2],card[3],card[0]);}
```

```
                    fast1(ar,as);
                    }

if (x==2) {calculate (card[0],card[1],card[2],card[3],card[4]);
          if ((hold[0]==1) and (hold[1]==1)) {fast2(ar,as,br,bs);
               if ((ar==br) and (ar>8)) {holdrank=1;}
                 else if (ar==br) {holdrank=4;}}
          if ((hold[0]==1) and (hold[2]==1)) {fast2(ar,as,cr,cs);
               if ((ar==cr) and (ar>8)) {holdrank=1;}
                 else if (ar==cr) {holdrank=4;}}
          if ((hold[0]==1) and (hold[3]==1)) {fast2(ar,as,dr,ds);
               if ((ar==dr) and (ar>8)) {holdrank=1;}
                 else if (ar==dr) {holdrank=4;}}
          if ((hold[0]==1) and (hold[4]==1)) {fast2(ar,as,er,es);
               if ((ar==er) and (ar>8)) {holdrank=1;}
                    else if (ar==er) {holdrank=4;}}
          if ((hold[1]==1) and (hold[2]==1)) {fast2(br,bs,cr,cs);
               if ((br==cr) and (br>8)) {holdrank=1;}
                    else if (br==cr) {holdrank=4;}}
          if ((hold[1]==1) and (hold[3]==1)) {fast2(br,bs,dr,ds);
               if ((br==dr) and (br>8)) {holdrank=1;}
                    else if (br==dr) {holdrank=4;}}
          if ((hold[1]==1) and (hold[4]==1)) {fast2(br,bs,er,es);
               if ((br==er) and (br>8)) {holdrank=1;}
                    else if (br==er) {holdrank=4;}}
          if ((hold[2]==1) and (hold[3]==1)) {fast2(cr,cs,dr,ds);
               if ((cr==dr) and (cr>8)) {holdrank=1;}
                    else if (cr==dr) {holdrank=4;}}
          if ((hold[2]==1) and (hold[4]==1)) {fast2(cr,cs,er,es);
               if ((cr==er) and (cr>8)) {holdrank=1;}
                    else if (cr==er) {holdrank=4;}}
          if ((hold[3]==1) and (hold[4]==1)) {fast2(dr,ds,er,es);
               if ((dr==er) and (dr>8)) {holdrank=1;}
                    else if (dr==er) {holdrank=4;}}
        }


if (x==3) {calculate (card[0],card[1],card[2],card[3],card[4]);
          if ((hold[0]==0) and (hold[1]==0)) {fast3(cr,cs,dr,ds,er,es);
               if ((cr==dr) and (dr==er)) {holdrank=2;}
               if (((cr==dr) and (cr>8)) or ((dr==er) and (dr>8))
             or ((cr==er) and (cr>8))) {holdjs=1;}}

          if ((hold[0]==0) and (hold[2]==0)) {fast3(br,bs,dr,ds,er,es);
               if ((br==dr) and (dr==er)) {holdrank=2;}
               if (((br==dr) and (br>8)) or ((dr==er) and (dr>8)) or
```

```
                    ((br==er) and (br>8))) {holdjs=1;}}

                if ((hold[0]==0) and (hold[3]==0)) {fast3(br,bs,cr,cs,er,es);
                    if ((br==cr) and (cr==er)) {holdrank=2;}
                     if (((br==cr) and (br>8)) or ((cr==er) and (cr>8)) or
                            ((br==er) and (br>8))) {holdjs=1;}}

                if ((hold[0]==0) and (hold[4]==0)) {fast3(br,bs,cr,cs,dr,ds);
                    if ((br==cr) and (cr==dr)) {holdrank=2;}
                     if (((br==cr) and (br>8)) or ((br==dr) and (br>8)) or
                            ((cr==dr) and (cr>8))) {holdjs=1;}}

                if ((hold[1]==0) and (hold[2]==0)) {fast3(ar,as,dr,ds,er,es);
                    if ((ar==dr) and (dr==er)) {holdrank=2;}
                     if (((ar==dr) and (ar>8)) or ((dr==er) and (dr>8)) or
                            ((ar==er) and (ar>8))) {holdjs=1;}}

                if ((hold[1]==0) and (hold[3]==0)) {fast3(ar,as,cr,cs,er,es);
                    if ((ar==cr) and (cr==er)) {holdrank=2;}
                            if (((ar==cr) and (ar>8)) or ((cr==er) and (cr>8)) or
                                    ((ar==er) and (ar>8))) {holdjs=1;}}

                if ((hold[1]==0) and (hold[4]==0)) {fast3(ar,as,cr,cs,dr,ds);
                    if ((ar==cr) and (cr==dr)) {holdrank=2;}
                     if (((ar==cr) and (ar>8)) or ((ar==dr) and (ar>8)) or
                            ((cr==dr) and (cr>8))) {holdjs=1;}}

                if ((hold[2]==0) and (hold[3]==0)) {fast3(ar,as,br,bs,er,es);
                    if ((ar==br) and (br==er)) {holdrank=2;}
                     if (((ar==br) and (ar>8)) or ((br==er) and (br>8)) or
                            ((ar==er) and (ar>8))) {holdjs=1;}}

                if ((hold[2]==0) and (hold[4]==0)) {fast3(ar,as,br,bs,dr,ds);
                    if ((ar==br) and (br==dr)) {holdrank=2;}
                     if (((ar==br) and (ar>8)) or ((br==dr) and (br>8)) or
                            ((ar==dr) and (ar>8))) {holdjs=1;}}

                if ((hold[3]==0) and (hold[4]==0)) {fast3(ar,as,br,bs,cr,cs);
                    if ((ar==br) and (br==cr)) {holdrank=2;}
                     if (((ar==br) and (ar>8)) or ((br==cr) and (br>8)) or
                            ((ar==cr) and (ar>8))) {holdjs=1;}}
        }

if (x==4) {calculate (card[0],card[1],card[2],card[3],card[4]);

        if (hold[0]==0) {slow4 (card[1],card[2],card[3],card[4]);
```

```
                    if ((br==cr) and (cr==dr) and (dr==er)) {holdrank=5;}
                     else
                    if (((br==cr) and (dr==er)) or
                            ((br==dr) and (cr==er)) or
                             ((br==er) and (cr==dr))) {holdrank=3;}
                    if (((br==cr) and (cr==dr)) or ((cr==dr) and (dr==er))
                                    or ((br==cr) and (cr==er)) or ((br==dr) and
(dr==er)))
                                    {hold3=1;}

                                if (((br==cr) and (br>8)) or ((cr==dr) and (cr>8))
                                    or ((dr==er) and (dr>8))) {hold2=1;}}


                if (hold[1]==0) {slow4 (card[0],card[2],card[3],card[4]);
                if ((ar==cr) and (cr==dr) and (dr==er)) {holdrank=5;}
                      else if (((ar==cr) and (dr==er)) or
                            ((ar==dr) and (cr==er)) or
                             ((ar==er) and (cr==dr))) {holdrank=3;}
                      if (((ar==cr) and (cr==dr)) or ((cr==dr) and (dr==er))
                                    or ((ar==cr) and (cr==er)) or ((ar==dr) and
(dr==er)))
                                    {hold3=1;}
                                if (((ar==cr) and (ar>8)) or ((cr==dr) and (cr>8))
                                    or ((dr==er) and (dr>8))) {hold2=1;}}

                if (hold[2]==0) {slow4 (card[0],card[1],card[3],card[4]);
                if ((ar==br) and (br==dr) and (dr==er)) {holdrank=5;}
                      else if (((ar==br) and (dr==er)) or
                            ((ar==dr) and (br==er)) or
                             ((ar==er) and (br==dr))) {holdrank=3;}
                     if (((ar==br) and (br==dr)) or ((br==dr) and (dr==er))
                                    or ((ar==br) and (br==er)) or ((ar==dr) and
(dr==er)))
                                    {hold3=1;}
                                  if (((ar==br) and (ar>8)) or ((br==dr) and (br>8))
                                   or ((dr==er) and (dr>8))) {hold2=1;}}

                if (hold[3]==0) {slow4 (card[0],card[1],card[2],card[4]);
                if ((ar==br) and (br==cr) and (cr==er)) {holdrank=5;}
                      else if (((ar==br) and (cr==er)) or
                            ((ar==cr) and (br==er)) or
                             ((ar==er) and (br==cr))) {holdrank=3;}
                     if (((ar==br) and (br==cr)) or ((br==cr) and (cr==er))
                                    or ((ar==br) and (br==er)) or ((ar==cr) and
(cr==er)))
```

```
                                {hold3=1;}
                            if (((ar==br) and (ar>8)) or ((br==cr) and (br>8))
                                or ((cr==er) and (er>8))) {hold2=1;}}

                    if (hold[4]==0) {slow4 (card[0],card[1],card[2],card[3]);
                     if ((ar==br) and (br==cr) and (cr==dr)) {holdrank=5;}
                          else if (((ar==br) and (cr==dr)) or
                               ((ar==cr) and (br==dr)) or
                               ((ar==dr) and (br==cr))) {holdrank=3;}
                        if (((ar==br) and (br==cr)) or ((br==cr) and (cr==dr))
                                or ((ar==br) and (br==dr)) or ((ar==cr) and
(cr==dr)))

                                {hold3=1;}
                            if (((ar==br) and (ar>8)) or ((br==cr) and (br>8))
                                or ((cr==dr) and (dr>8))) {hold2=1;}}
                    }

                    if (x==5) {total[1]=0; total[2]=0; total[3]=0; total[4]=0;
                          total[5]=0; total[6]=0; total[7]=0; total[8]=0;
                                total[9]=0;
                                    removemovieclip (redDDonid);
                                    reddd=0;}

                        // holding 4 of a kind, need to deactivate double button
                        if ((x==4) and
                              (((hold[0]==0) and ((br==cr) and (cr==dr) and (dr==er)))
or

                           ((hold[1]==0) and ((ar==cr) and (cr==dr) and (dr==er))) or
                           ((hold[2]==0) and ((ar==br) and (br==dr) and (dr==er))) or
                           ((hold[3]==0) and ((ar==br) and (br==cr) and (cr==er))) or
                                    ((hold[4]==0) and ((ar==br) and (br==cr) and
(cr==dr)))))

                                {total[1]=0; total[2]=0; total[3]=0; total[4]=0;
                          total[5]=0; total[6]=0; total[7]=0; total[8]=0;
                                total[9]=0;
                                    removemovieclip (redDDonid);
                                    reddd=0;}

                    if ((x!=5) and (reddd==0) and (gameover==0)) {
                _root.attachmovie ("redDDon","redDDonid",++depth);
                _root["redDDonid"]._x=buttonDDx;
                _root["redDDonid"]._y=buttonDDy;
                reddd=1;
                _root["redDDonid"].onRelease=function() {
                        removemovieclip("redDDonid");
                        removemovieclip("reddealdrawonid");
```

```
        reddd=0;
        double=2;
        bets=coins*double
        _root.doubleinfobox.htmltext="<BR>  DOUBLED!";
        _root.coinsbetbox.htmltext=bets;
        money=money-coins;
        _root.moneybox.htmltext='<p align="center">$'+money;
        playclick();
        for (i=0; i<5; ++i) {
                var cardid="card"+card[i]+"big"+"id";
                if (_root[cardid]._held==0) {
                        attachmovie ("wiz","wiz"+i,++depth);
                        _root["wiz"+i]._x=cardsx+i*150;
                        _root["wiz"+i]._y=cardsy;}
        }
                cardCount1=-1;
                done=0;
                intervalID1=setInterval(drawone, delay);
}}


        p1=total[9]/total[10];
        p2=total[8]/total[10];
        p3=total[7]/total[10];
        p4=total[6]/total[10];
        p5=total[5]/total[10];
        p6=total[4]/total[10];
        p7=total[3]/total[10];
        p8=total[2]/total[10];
        p9=total[1]/total[10];

        payinghands=0;
        if (total[1]!=0) {payinghands++};
        if (total[2]!=0) {payinghands++};
        if (total[3]!=0) {payinghands++};
        if (total[4]!=0) {payinghands++};
        if (total[5]!=0) {payinghands++};
        if (total[6]!=0) {payinghands++};
        if (total[7]!=0) {payinghands++};
        if (total[8]!=0) {payinghands++};
        if (total[9]!=0) {payinghands++};


        r1=0; r2=0; r3=0; r4=0; r5=0; r6=0; r7=0; r8=0; r9=0;
        if (p1!=0) {r1=((coins*gameret)/p1)/payinghands;}
        if (p2!=0) {r2=((coins*gameret)/p2)/payinghands;}
```

```
                if (p3!=0) {r3=((coins*gameret)/p3)/payinghands;}
                if (p4!=0) {r4=((coins*gameret)/p4)/payinghands;}
                if (p5!=0) {r5=((coins*gameret)/p5)/payinghands;}
                if (p6!=0) {r6=((coins*gameret)/p6)/payinghands;}
                if (p7!=0) {r7=((coins*gameret)/p7)/payinghands;}
                if (p8!=0) {r8=((coins*gameret)/p8)/payinghands;}
                if (p9!=0) {r9=((coins*gameret)/p9)/payinghands;}

p[1]=p1; p[2]=p2; p[3]=p3; p[4]=p4; p[5]=p5; p[6]=p6; p[7]=p7; p[8]=p8; p[9]=p9;
pay[1]=r1; pay[2]=r2; pay[3]=r3; pay[4]=r4; pay[5]=r5;
pay[6]=r6; pay[7]=r7; pay[8]=r8; pay[9]=r9;


        // player is dealt J or better and holds the pair, have to 0 out rank 9 payout
                if (holdrank==1)
{f=(pay[7]*p[7]+pay[8]*p[8]+pay[9]*p[9])/(pay[7]*p[7]+pay[8]*p[8]);
                                                pay[8]=pay[8]*f;
                                                pay[7]=pay[7]*f;
                                                pay[9]=0;
                                                payinghands--;
                // now lets shift some down for 4oak to ranks 7 and 8
                                                s=.5;  //shrinking factor for 4oak
                                                f=(pay[3]*p[3]+pay[8]*p[8]+pay[7]*p[7]-
s*pay[3]*p[3])/(pay[7]*p[7]+pay[8]*p[8]);
                                                pay[3]=pay[3]*s;
                                                pay[7]=pay[7]*f;
                                                pay[8]=pay[8]*f;
                                                }

                // player is dealt 3 of a kind, need to shift to full house and 4oak
                if (holdrank==2) {f=coins/(coins-pay[7]*p[7]);

                                                pay[4]=pay[4]*f;
                                                pay[3]=pay[3]*f;
                                                pay[7]=0;
                }

                // player is dealt 2 pair, ned to shift to full house
                if (holdrank==3) {f=coins/(coins-pay[8]*p[8]);
                                                        pay[4]=pay[4]*f;
                                                        pay[8]=0;
                                                }

                // player is dealt a low pair, so shift payouts from top

                if (holdrank==4) {s=.4;  //shrinking factor for 4oak
```

```
                                        f=(pay[3]*p[3]+pay[8]*p[8]+pay[7]*p[7]-
s*pay[3]*p[3])/(pay[7]*p[7]+pay[8]*p[8]);

                                        pay[3]=pay[3]*s;
                                        pay[7]=pay[7]*f;
                                        pay[8]=pay[8]*f;
                                        s=.75;  //lets take some off full house too
                                        f=(pay[4]*p[4]+pay[8]*p[8]+pay[7]*p[7]-
s*pay[4]*p[4])/(pay[7]*p[7]+pay[8]*p[8]);

                                        pay[4]=pay[4]*s;
                                        pay[7]=pay[7]*f;
                                        pay[8]=pay[8]*f;
                                        }
```

```
// if J or better pays less than coins

for (i=1; i<6; i++) {
if ((pay[9]>0) and (pay[9]<coins)) {
        s=.75;
        f=(pay[1]*p[1]+pay[9]*p[9]-s*pay[1]*p[1])/(pay[9]*p[9]);
        pay[1]=pay[1]*s;
        pay[9]=pay[9]*f;
}


if ((pay[9]>0) and (pay[9]<coins)) {
        s=.75;
        f=(pay[2]*p[2]+pay[9]*p[9]-s*pay[2]*p[2])/(pay[9]*p[9]);
        pay[2]=pay[2]*s;
        pay[9]=pay[9]*f;
}

if ((pay[9]>0) and (pay[9]<coins)) {
        s=.75;
        f=(pay[3]*p[3]+pay[9]*p[9]-s*pay[3]*p[3])/(pay[9]*p[9]);
        pay[3]=pay[3]*s;
        pay[9]=pay[9]*f;}

        if ((pay[9]>0) and (pay[9]<coins)) {
        s=.9;
        f=(pay[4]*p[4]+pay[9]*p[9]-s*pay[4]*p[4])/(pay[9]*p[9]);
        pay[4]=pay[4]*s;
        pay[9]=pay[9]*f;
```

```
}
}

// ok now try using the lower payouts to make rank 9 >= coins

for (i=1; i<6; i++) {
if ((pay[9]>0) and (pay[9]<coins)) {
        s=.75;
        f=(pay[4]*p[4]+pay[9]*p[9]-s*pay[4]*p[4])/(pay[9]*p[9]);
        pay[4]=pay[4]*s;
        pay[9]=pay[9]*f;
}


if ((pay[9]>0) and (pay[9]<coins)) {
        s=.75;
        f=(pay[5]*p[5]+pay[9]*p[9]-s*pay[5]*p[5])/(pay[9]*p[9]);
        pay[5]=pay[5]*s;
        pay[9]=pay[9]*f;
}

if ((pay[9]>0) and (pay[9]<coins)) {
        s=.75;
        f=(pay[6]*p[6]+pay[9]*p[9]-s*pay[6]*p[6])/(pay[9]*p[9]);
        pay[6]=pay[6]*s;
        pay[9]=pay[9]*f;}

        if ((pay[9]>0) and (pay[9]<coins)) {
        s=.9;
        f=(pay[7]*p[7]+pay[9]*p[9]-s*pay[7]*p[7])/(pay[9]*p[9]);
        pay[7]=pay[7]*s;
        pay[9]=pay[9]*f;
}

if ((pay[9]>0) and (pay[9]<coins)) {
        s=.75;
        f=(pay[8]*p[8]+pay[9]*p[9]-s*pay[8]*p[8])/(pay[9]*p[9]);
        pay[8]=pay[8]*s;
        pay[9]=pay[9]*f;}
}



// cap royal payout and transfer to bottom
royalcap1=(royalcap/5)*coins;
if (pay[1]>royalcap1) {
```

```
            s=royalcap1/pay[1];
            bottom=0;
            for (i=1; i<10; i++) {if (pay[i]!=0) {bottom=i;}}

f=(pay[bottom]*p[bottom]-s*pay[1]*p[1]+pay[1]*p[1])/(pay[bottom]*p[bottom]);
      pay[bottom]=f*pay[bottom];
      pay[1]=royalcap1;
            }

            // cap straight flush payout and transfer to bottom

            straightflushcap1=(straightflushcap/5)*coins;
      if (pay[2]>straightflushcap1) {
            s=straightflushcap1/pay[2];
            bottom=0;
            for (i=1; i<10; i++) {if (pay[i]!=0) {bottom=i;}}

      // if available, lets use 2 pair for bottom to balance things out
      if (pay[8]>0) {bottom=8;}

f=(pay[bottom]*p[bottom]-s*pay[2]*p[2]+pay[2]*p[2])/(pay[bottom]*p[bottom]);
      pay[bottom]=f*pay[bottom];
      pay[2]=straightflushcap1;
            }

            if ((holdrank!=2) and (holdjs==1)) {//shift J or better payout to 2 Pair

      f=(pay[8]*p[8]+pay[9]*p[9])/(pay[8]*p[8]);
                                          pay[8]=pay[8]*f;
                                          pay[9]=0;
            }


            //shift rank 7 to 4, player holds 3oak and misc card
            if ((holdrank!=5) and (hold3==1)) {
                  f=(pay[4]*p[4]+pay[7]*p[7])/(pay[4]*p[4]);
                                          pay[4]=pay[4]*f;
                                          pay[7]=0;
            }

            // holding 4 cards with high pair, need to shift 9 to 8
            if ((x==4) and (hold2==1) and (hold3!=1) and (holdrank!=3)) {
                  f=(pay[8]*p[8]+pay[9]*p[9])/(pay[8]*p[8]);
                                          pay[8]=pay[8]*f;
                                          pay[9]=0;
            }
```

```
//need to shift fractions up
bott=10;
findbottom();
for (i=1; i<10; ++i) {
bott=next;
findbottom();
// need to transfer fraction on pay[bott] to pay[next]
if (next!=0) {
want=math.floor(pay[bott]); want1=want;
if ((want>30) and (want <125)) {want=(coins*math.floor(want/coins));
        if (((want1/coins)-math.floor(want1/coins)) >.5) want=want+coins+.001;
}
else if ((want>125) and (want<1000)){want=(10*math.floor(want/10));
        if (((want1/10)-math.floor(want1/10)) >.5) want=want+10.001;}
else if ((want>25) and (want<30)) {want=30.01;}
else if ((want>1000)) {want=(50*math.floor(want/50));
        if (((want1/50)-math.floor(want1/50)) >.5) want=want+50.001;}
else if ((want>10000)) {want=(100*math.floor(want/100));
        if (((want1/100)-math.floor(want1/100)) >.5) want=want+100.001;}

s=want/pay[bott];
f=(pay[next]*p[next]+pay[bott]*p[bott]-s*pay[bott]*p[bott])/(pay[next]*p[next]);
pay[next]=f*pay[next];
pay[bott]=pay[bott]*s;
} else {i=9;}
}

findhighest();
if (bott!=highest) {
// transfer fractional part of bott to highest to improve accuracy
        s=math.floor(pay[bott])/pay[bott];
        f=(pay[highest]*p[highest]+pay[bott]*p[bott]-
s*pay[bott]*p[bott])/(pay[highest]*p[highest]);
pay[highest]=f*pay[highest];
pay[bott]=pay[bott]*s;
}

// straight flush may be bigger than cap again, lets move to royal if possible
if ((pay[2]>straightflushcap) and (pay[1]>0)) {
                s=straightflushcap/pay[2];
                bottom=1;
        f=(pay[bottom]*p[bottom]-
s*pay[2]*p[2]+pay[2]*p[2])/(pay[bottom]*p[bottom]);
        pay[bottom]=f*pay[bottom];
```

```
        pay[2]=straightflushcap;
}


if (x==5) {pay[1]=0; pay[2]=0; pay[3]=0; pay[4]=0; pay[5]=0; pay[6]=0;
    pay[7]=0; pay[8]=0; pay[9]=0;}

pay[9]=pay[9]+.00001

// round off large payouts by coins
for (i=1; i<10; ++i) {
        if (pay[i]>2000) {pay[i]=(coins*math.floor(pay[i]/coins));}
         else {pay[i]=math.floor(pay[i]);}
        }

                if (gameover==0) {
                _root.paytable2rank1.text=pay[1];
                _root.paytable2rank2.text=pay[2];
                _root.paytable2rank3.text=pay[3];
                _root.paytable2rank4.text=pay[4];
                _root.paytable2rank5.text=pay[5];
                _root.paytable2rank6.text=pay[6];
                _root.paytable2rank7.text=pay[7];
                _root.paytable2rank8.text=pay[8];
                _root.paytable2rank9.text=pay[9];

adv=p[1]*pay[1]+p[2]*pay[2]+p[3]*pay[3]+p[4]*pay[4]+p[5]*pay[5]+p[6]*pay[6]+p[7]
*pay[7]+p[8]*pay[8]+p[9]*pay[9];
                adv=adv/coins
                _root.returnbox.text="return="+adv;

                if ((x==4) and (adv==0)) {removemovieclip (redDDonid);
                                          reddd=0;}

                }

}


function showpays() {

//_root.rank1.htmltext='<FONT  COLOR="#00FF00">ROYAL FLUSH';
highlight='<FONT  COLOR="#ff0000">';
defaul='<FONT  COLOR="#fdfe7e">'
light[0]=defaul; light[1]=defaul; light[2]=defaul;light[3]=defaul;light[4]=defaul;
light[5]=defaul;light[6]=defaul; light[7]=defaul; light[8]=defaul; light[9]=defaul;
```

```
light[winningrank]=highlight;

bets=coins*double
if (double==1) {_root.doubleinfobox.htmltext="doubling pays <br>an additional";
                                _root.coinsinbox2.htmltext="          ";}
 else if (double==2) {//_root.doubleinfobox.htmltext="DOUBLED!";

 }

_root.rank1.htmltext=light[1]+ranks[1];
_root.rank2.htmltext=light[2]+ranks[2];
_root.rank3.htmltext=light[3]+ranks[3];
_root.rank4.htmltext=light[4]+ranks[4];
_root.rank5.htmltext=light[5]+ranks[5];
_root.rank6.htmltext=light[6]+ranks[6];
_root.rank7.htmltext=light[7]+ranks[7];
_root.rank8.htmltext=light[8]+ranks[8];
_root.rank9.htmltext=light[9]+ranks[9];

defaul='<p align="center"> <FONT  COLOR="#fdfe7e">';
light[0]=defaul; light[1]=defaul; light[2]=defaul;light[3]=defaul;light[4]=defaul;
light[5]=defaul;light[6]=defaul; light[7]=defaul; light[8]=defaul; light[9]=defaul;
light[winningrank]=defaul+highlight;

_root.paytable1rank1.htmltext=light[1]+payout[1]*coins;
_root.paytable1rank2.htmltext=light[2]+payout[2]*coins;
_root.paytable1rank3.htmltext=light[3]+payout[3]*coins;
_root.paytable1rank4.htmltext=light[4]+payout[4]*coins;
_root.paytable1rank5.htmltext=light[5]+payout[5]*coins;
_root.paytable1rank6.htmltext=light[6]+payout[6]*coins;
_root.paytable1rank7.htmltext=light[7]+payout[7]*coins;
_root.paytable1rank8.htmltext=light[8]+payout[8]*coins;
_root.paytable1rank9.htmltext=light[9]+payout[9]*coins;


if (double==1) {light[winningrank]=defaul;}
                _root.paytable2rank1.htmltext=light[1]+pay[1];
                _root.paytable2rank2.htmltext=light[2]+pay[2];
                _root.paytable2rank3.htmltext=light[3]+pay[3];
                _root.paytable2rank4.htmltext=light[4]+pay[4];
                _root.paytable2rank5.htmltext=light[5]+pay[5];
                _root.paytable2rank6.htmltext=light[6]+pay[6];
                _root.paytable2rank7.htmltext=light[7]+pay[7];
                _root.paytable2rank8.htmltext=light[8]+pay[8];
                _root.paytable2rank9.htmltext=light[9]+pay[9];
```

```
winningrank=0;
}


function shuffle()
{
for(var i=0;i<52;++i)
{card[i]=i+1;}
var randcard;
var temp;
for (var i=0; i<52; ++i) {
        randcard=math.floor (math.random()*51 +1);
        temp=card[randcard];
        card[randcard]=card[i];
        card[i]=temp;

}

//card[0]=13; card[1]=11;card[2]=12; card[3]=9; card[4]=10;
//card[0]=12; card[1]=25; card[2]=38;
//card[0]=1;card[1]=14;card[2]=27;card[4]=40;
//card[0]=1;card[1]=14;card[2]=2;card[3]=15;
//card[0]=10; card[1]=2; card[2]=3; card[3]=4; card[4]=5; card[10]=1;

}

function playcard()
{
mySound=new Sound(this);
mySound.attachSound("cardturn");
mySound.setVolume(200);
mySound.start();
}

function playclick()
{
mySound=new Sound(this);
mySound.attachSound("click");
mySound.setVolume(100);
mySound.start();
}

function playwin0()
```

```
{
mySound=new Sound(this);
mySound.attachSound("smallwin");
mySound.setVolume(100);
mySound.start();
}


function playwin()
{
if (chimes==0) {clearinterval(intervalchime); return;}
else {chimes--;
mySound=new Sound(this);
mySound.attachSound("smallwin");
mySound.setVolume(100);
mySound.start();
}
}

function playdiamond()
{
mySound=new Sound(this);
mySound.attachSound("diamond");
mySound.setVolume(100);
mySound.start();
}

function playbigwin()
{
mySound=new Sound(this);
mySound.attachSound("bigwin");
mySound.setVolume(100);
mySound.start();
}



function dealblankcards() {
        for (i=0; i<5; ++i) {
                removemovieclip ("wiz"+i);
                attachmovie ("wiz","wiz"+i,++depth);
                _root["wiz"+i]._x=cardsx+i*150;
                _root["wiz"+i]._y=cardsy;
        }
}
```

```
function payhand (a,b,c,d,e)
//determine the rank of the final hand
{
        lastcard=[];
        lastcard[1]=a;lastcard[2]=b;lastcard[3]=c;lastcard[4]=d;lastcard[5]=e;
        finalvalue=[];
        finalsuit=[];
        for (i=1; i<6; ++i) {
        cardindex=lastcard[i];
        suit=1
        if (cardindex>13) { cardindex=cardindex-13; ++suit;}
        if (cardindex>13) { cardindex=cardindex-13;++suit;}
        if (cardindex>13) { cardindex=cardindex-13;++suit;}
        returncard=cardindex;

        finalvalue[i]=returncard;
        finalsuit[i]=suit;}


        for (j=1;j<5;++j){
        for (i=1;i<5;++i) {
                if (finalvalue[i]>finalvalue[i+1]) {tempvalue=finalvalue[i];

tempsuit=finalsuit[i];

        finalvalue[i]=finalvalue[i+1];

        finalsuit[i]=finalsuit[i+1];

        finalvalue[i+1]=tempvalue;

        finalsuit[i+1]=tempsuit;
                }
        }}

        flush=0; fourkind=0; straight=0; fullhouse=0; threekind=0; twopair=0;
jorbetter=0; rank=0;



        if ((finalsuit[1]==finalsuit[2]) and (finalsuit[2]==finalsuit[3]) and
                        (finalsuit[3]==finalsuit[4]) and (finalsuit[4]==finalsuit[5]))
{flush=1;}

        if ((finalvalue[1]==finalvalue[2]) and (finalvalue[2]==finalvalue[3]) and
```

```
                (finalvalue[3]==finalvalue[4])) {fourkind=1;}

                if ((finalvalue[2]==finalvalue[3]) and (finalvalue[3]==finalvalue[4]) and
                (finalvalue[4]==finalvalue[5])) {fourkind=1;}

            if ((finalvalue[1]==finalvalue[2]) and (finalvalue[2]==finalvalue[3]) and
                        (finalvalue[4]==finalvalue[5])) {fullhouse=1;}


            if ((finalvalue[3]==finalvalue[4]) and (finalvalue[4]==finalvalue[5]) and
                        (finalvalue[1]==finalvalue[2])) {fullhouse=1;}

            if ((finalvalue[1]==finalvalue[2]-1) and (finalvalue[2]==finalvalue[3]-1) and
                    (finalvalue[3]==finalvalue[4]-1) and (finalvalue[4]==finalvalue[5]-1))
                            {straight=1;}

            if ((finalvalue[1]==1) and (finalvalue[2]==10) and (finalvalue[3]==11) and
                        (finalvalue[4]==12) and (finalvalue[5]==13)) {straight=1;}

             if (((finalvalue[1]==finalvalue[2]) and (finalvalue[2]==finalvalue[3])) or
           ((finalvalue[2]==finalvalue[3]) and (finalvalue[3]==finalvalue[4])) or
                  ((finalvalue[3]==finalvalue[4]) and (finalvalue[4]==finalvalue[5])))
                              {threekind=1;}

              if (((finalvalue[1]==finalvalue[2]) and (finalvalue[3]==finalvalue[4])) or
                 ((finalvalue[2]==finalvalue[3]) and (finalvalue[4]==finalvalue[5]))
                       or ((finalvalue[1]==finalvalue[2]) and (finalvalue[4]==finalvalue[5])))
                         {twopair=1;}

             if (((finalvalue[1]==finalvalue[2]) and (finalvalue[1]>10 or finalvalue[1]==1))
or
                    ((finalvalue[2]==finalvalue[3]) and (finalvalue[2]>10 or
finalvalue[2]==1)) or
                    ((finalvalue[3]==finalvalue[4]) and (finalvalue[3]>10 or
finalvalue[3]==1)) or
                    ((finalvalue[4]==finalvalue[5]) and (finalvalue[4]>10 or
finalvalue[4]==1)))
                {jorbetter=1;}

            if (fourkind==1) {rank=3;}
             else if ((straight==1) and (flush==1) and (finalvalue[2]==10)
                                        and (finalvalue[1]!=9)) {rank=1;}
                  else if ((straight==1) and (flush==1)) {rank=2;}
                        else if (fullhouse==1) {rank=4;}
                              else if (flush==1) {rank=5;}
                                 else if (straight==1) {rank=6;}
```

```
                                else if (threekind==1) {rank=7;}
                                        else if (twopair==1) {rank=8;}
                                                else if (jorbetter==1) {rank=9;}
}


function dealinitialcards()
{
                _root.coinsbetbox.htmltext=coins;
                if(cardCount>=4) {
                clearInterval(intervalID);
                payhand (card[0],card[1],card[2],card[3],card[4]);
                winningrank=rank;
                showpays();
                adjusttable2();
        _root.attachmovie ("reddealdrawon","reddealdrawonid",++depth);
        _root["reddealdrawonid"]._x=buttondrawx;
        _root["reddealdrawonid"]._y=buttondrawy;
        _root["reddealdrawonid"].onRelease=function() {
                removemovieclip("reddealdrawonid");
                double=1;
                playclick();
                clearinterval(intervalIDM);
                money=money1;
                _root.moneybox.htmltext='<p align="center">$'+money;
                for (i=0; i<5; ++i) {
                        var cardid="card"+card[i]+"big"+"id";
                        if (_root[cardid]._held==0) {
                                attachmovie ("wiz","wiz"+i,++depth);
                                _root["wiz"+i]._x=cardsx+i*150;
                                _root["wiz"+i]._y=cardsy;}
                }

                        cardCount1=-1;
                        done=0;
                        intervalID1=setInterval(drawone, delay);

        }


        _root.attachmovie ("redDDon","redDDonid",++depth);
        _root["redDDonid"]._x=buttonDDx;
        _root["redDDonid"]._y=buttonDDy;
        reddd=1;
        _root["redDDonid"].onRelease=function() {
                removemovieclip("redDDonid");
```

```
                removemovieclip("reddealdrawonid");
                reddd=0;
                double=2;
                bets=coins*double
                _root.doubleinfobox.htmltext="<BR>  DOUBLED!";
                _root.coinsbetbox.htmltext=bets;
                money=money-coins;
                _root.moneybox.htmltext='<p align="center">$'+money;
                playclick();
                for (i=0; i<5; ++i) {
                        var cardid="card"+card[i]+"big"+"id";
                        if (_root[cardid]._held==0) {
                                attachmovie ("wiz","wiz"+i,++depth);
                                _root["wiz"+i]._x=cardsx+i*150;
                                _root["wiz"+i]._y=cardsy;}
                }

                        cardCount1=-1;
                        done=0;
                        intervalID1=setInterval(drawone, delay);

        }



                for (i=0; i<5; ++i) {
                removemovieclip ("wiz"+i);}
                return; }

        ++cardCount;
        hold[cardcount]=0;
        var cardname="card"+card[cardcount]+"big";
var cardid=cardname+"id";
playcard();
_root.attachMovie(cardname, cardid, ++depth);
_root[cardid]._x=cardsx+(cardcount*150);
_root[cardid]._y=cardsy;
_root[cardid]._held=0;


_root[cardid].onRelease=function() {

                if (this._name=="card"+card[0]+"bigid") {heldcard=0;}
                if (this._name=="card"+card[1]+"bigid") {heldcard=1;}
                if (this._name=="card"+card[2]+"bigid") {heldcard=2;}
          if (this._name=="card"+card[3]+"bigid") {heldcard=3;}
```

```
            if (this._name=="card"+card[4]+"bigid") {heldcard=4;}

                if (gameover==0) {
                        playclick();
                        heldid="held"+this._name;
                        if (_root[this._name]._held==0) {
                        _root[this._name]._held=1;
                        hold[heldcard]=1;
                        _root.attachMovie("held", heldid, ++depth);
                        _root[heldid]._x=_root[this._name]._x;
                        _root[heldid]._y=_root[this._name]._y;}
                                else {removeMovieClip(heldid);
                                _root[this._name]._held=0;
                                hold[heldcard]=0;}
                        }

                adjusttable2();
        }
}

function raisemoney()
{
        if (money==money1) {clearinterval(intervalIDM); return;}
        money++;
        _root.moneybox.htmltext='<p align="center">$'+money;
}

function drawone()
{
        ++cardcount1;
        removemovieclip("redDDonid");
        reddd=1;
        var cardid="card"+card[cardcount1]+"big"+"id";
        if (_root[cardid]._held==1) {cardcount1++;}
        var cardid="card"+card[cardcount1]+"big"+"id";
        if (_root[cardid]._held==1) {cardcount1++;}
        var cardid="card"+card[cardcount1]+"big"+"id";
        if (_root[cardid]._held==1) {cardcount1++;}
        var cardid="card"+card[cardcount1]+"big"+"id";
        if (_root[cardid]._held==1) {cardcount1++;}

        if(cardCount1>=5) {
                clearInterval(intervalID1);

                { gameover=1;
                        payhand (card[0],card[1],card[2],card[3],card[4]);
```

```
                    winningrank=rank;
                    showpays();
                    if (rank==1) {playbigwin();}
                    chimes=0;
                    if (rank>1) {playwin0();}
                    if ((rank>1) and (double==2) and (pay[rank]>0)) {chimes=1;
                    intervalchime=setinterval(playwin,600);}
                    money1=money+payout[rank]*coins;
                    if (double==2) {money1=money1+pay[rank];}
                    intervalIDM=setInterval(raisemoney, 25);
//                  _root.moneybox.text='<p align="center">$'+money;
                    totalwin=payout[rank]*coins;
                    if (double==2) {totalwin=totalwin+pay[rank];}
                    _root.winbox.htmltext='<p align="center">$'+totalwin;
                    startgame();
                    return;
                }
                return;}

                        var cardname="card"+card[cardcount1]+"big";
                        var cardid=cardname+"id";
                        if (_root[cardid]._held==0) {
                            smallcard="card"+card[cardcount1]+"small";
                            discards++;
                            discardids[discards]=smallcard+"id";

        //_root.attachmovie(smallcard,smallcard+"id",++depth);
                            //_root[smallcard+"id"]._x=125+discards*75;
                            //_root[smallcard+"id"]._y=discardsy;

                            removemovieclip(cardname+"id");
                            card[cardcount1]=card[10+cardcount1];
                            var card1name="card"+card[10+cardcount1]+"big";
                            var card1id=card1name+"id";
                            playcard();
                            _root.attachmovie(card1name,card1id,++depth);
                            _root[card1id]._x=cardsx+(cardcount1*150);
                            _root[card1id]._y=cardsy;
                            _root[card1id]._held=0;
                }
}


function dealCards()
{
        dealblankcards();
```

```
                    cardCount=-1;
                    intervalID=setInterval(dealinitialcards, delay);
}

function begin() {
        removemovieclip (redbet1onid);
        removemovieclip (redbet5onid);
        money=money-coins; money1=money;
_root.moneybox.htmltext='<p align="center">$'+money;
winningrank=0;
showpays();

for (i=0; i<5; ++i) {
var cardname="card"+card[i]+"big";
var cardid=cardname+"id";
removemovieclip (cardid);
removemovieclip ("held"+cardid);
}

for (i=0; i<6; ++i) {
        removemovieclip (discardids[i]);}

shuffle();
discards=0;
gameover=0;
                _root.paytable2rank1.text="    ";
                _root.paytable2rank2.text="    ";
                _root.paytable2rank3.text="    ";
                _root.paytable2rank4.text="    ";
                _root.paytable2rank5.text="    ";
                _root.paytable2rank6.text="    ";
                _root.paytable2rank7.text="    ";
                _root.paytable2rank8.text="    ";
                _root.paytable2rank9.text="    ";
dealcards();
}

function startgame () {
        activateredbet1();
        activateredbet5();
attachmovie("reddealdrawon","reddealdrawonid",++depth);
_root["reddealdrawonid"]._x=buttondrawx;
_root["reddealdrawonid"]._y=buttondrawy;
_root["reddealdrawonid"].onRelease=function() {
removemovieclip ("reddealdrawonid");
```

```
clearinterval(intervalIDM);
money=money1;
_root.moneybox.htmltext='<p align="center">$'+money;
double=1;
_root.winbox.htmltext="    ";
_root.returnbox.text="            ";
playclick();
begin();
}
}

function activateredbet1() {
        _root.attachmovie ("redbet1on","redbet1onid",++depth);
        _root["redbet1onid"]._x=buttonbet1x;
        _root["redbet1onid"]._y=buttonbet1y;
        _root["redbet1onid"].onRelease=function() {
                playclick();
                _root.winbox.htmltext="    ";
                _root.coinsinbox2.htmltext="        ";
                coins++; double=1;
                if (coins==6) {coins=1;}
                _root.coinsbetbox.htmltext=coins;
                showpays();
                _root.paytable2rank1.htmltext="    ";
                _root.paytable2rank2.text="    ";
                _root.paytable2rank3.text="    ";
                _root.paytable2rank4.text="    ";
                _root.paytable2rank5.text="    ";
                _root.paytable2rank6.text="    ";
                _root.paytable2rank7.text="    ";
                _root.paytable2rank8.text="    ";
                _root.paytable2rank9.text="    ";
                _root.returnbox.text="            ";
        }
}

function activateredbet5() {
        _root.attachmovie ("redbet5on","redbet5onid",++depth);
        _root["redbet5onid"]._x=buttonbet5x;
        _root["redbet5onid"]._y=buttonbet5y;
        _root["redbet5onid"].onRelease=function() {
                playclick();
                _root.winbox.htmltext="    ";
                _root.coinsbetbox.htmltext=5;
                _root.coinsinbox2.htmltext="        ";
                coins=5; double=1;
```

```
                showpays();
                _root.paytable2rank1.htmltext="    ";
                _root.paytable2rank2.text="    ";
                _root.paytable2rank3.text="    ";
                _root.paytable2rank4.text="    ";
                _root.paytable2rank5.text="    ";
                _root.paytable2rank6.text="    ";
                _root.paytable2rank7.text="    ";
                _root.paytable2rank8.text="    ";
                _root.paytable2rank9.text="    ";
                _root.returnbox.text="          ";
        }
}


discardids=[]; hold=[];
card=[]; payout=[];ranks=[];light=[];
p=[]; pay=[];
payout[1]=800; payout[2]=50; payout[3]=25; payout[4]=9;
payout[5]=6; payout[6]=4; payout[7]=3; payout[8]=2; payout[9]=1;
money=1000; money1=money;
royalcap=20000; straightflushcap=15000;
gameret=.99;

ranks[1]="ROYAL FLUSH"; ranks[2]="STRAIGHT FLUSH"; ranks[3]="4 OF A
KIND";
ranks[4]="FULL HOUSE"; ranks[5]="FLUSH"; ranks[6]="STRAIGHT"; ranks[7]="3
OF A KIND";
ranks[8]="2 PAIR"; ranks[9]="JACKS OR BETTER";

cardsx=140; cardsy=425; discardsy=550; buttonx=450; buttony=600;
maxbuttonx=700; paytablex=455; paytabley=120; delay=150;

buttonbet1x=300; buttonbet1y=625;
buttonbet5x=401; buttonbet5y=625;
buttondrawx=502; buttondrawy=625; buttonDDx=603; buttonDDy=625;
coins=1;

_root.moneybox.htmltext='<p align="center">$'+money;

winningrank=0;
showpays();
dealblankcards();

attachmovie ("redbet1off","redbet1offid",++depth);
_root["redbet1offid"]._x=buttonbet1x;
```

```
_root["redbet1offid"]._y=buttonbet1y;
attachmovie ("redbet5off","redbet5offid",++depth);
_root["redbet5offid"]._x=buttonbet5x;
_root["redbet5offid"]._y=buttonbet5y;
attachmovie ("reddealdrawoff","reddealdrawoffid",++depth);
_root["reddealdrawoffid"]._x=buttondrawx;
_root["reddealdrawoffid"]._y=buttondrawy;
attachmovie ("redDDoff","redDDoffid",++depth);
_root["redDDoffid"]._x=buttonDDx;
_root["redDDoffid"]._y=buttonDDy;


startgame();
```